SOFT-LOGIC

Paylogic

Сценарии оплаты для усовершенствованного модуля ввода данных.

Программное обеспечение «Процессинговый центр Pay-logic».

Руководство пользователя

АННОТАЦИЯ

Описывает процесс создания и конфигурирования сервисов с использованием сценариев оплаты усовершенствованного модуля ввода данных программного обеспечения «Процессинговый центр Pay-logic»

Версия руководства: 8.20

Руководство актуально для кабинета «Процессингового центра Pay-logic» версий 5.9.x, ТПО версий 5.1xx, 7.1xx, РМА версий 6.6x, мобильных приложений приема платежей версий 3.3.x

ООО «Софт-Лоджик»

656006, г. Барнаул, Малахова ул., дом 146в

Тел: (3852) 72-27-27

ОГЛАВЛЕНИЕ

Web: https://soft-logic.ru/
Mail: info@soft-logic.ru

Web: https://soft-logic.ru/
Mail: info@soft-logic.ru

ИСТОРИЯ ИЗМЕНЕНИЙ

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 8.10

Дата публикации: 02.05.2023.

Изменение	Раздел
Улучшения в документации:	
Актуализирован раздел по голосовому сопровождению	10.1

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 8.11

Дата публикации: 26.05.2023.

Изменение	Раздел
Улучшения в версии 5.136:	
Добавлено маскирование номера карты в логах ТПО	5.5

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 8.12

Дата публикации: 17.06.2023.

Изменение	Раздел
Дополнения в документации:	
Добавлено описание атрибута **`next-regex`** поля ввода `<text-field>` на экране **`numeric`**	5.7

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 8.13

Дата публикации: 15.11.2023.

Изменение	Раздел
Улучшения в версии PMA 6.31:	
Реализована обработка параметра **`set-sum-limit`**	6.14

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 8.14

Дата публикации: 05.12.2023.

Изменение	Раздел
Улучшения в версии PMA 6.33:	
Реализована поддержка действия `<encode>`	6.4.4
Реализована поддержка действия `<load>`	6.17

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 8.15

Дата публикации: 26.11.2024.

Изменение	Раздел
Улучшения в документации:	
Внесены уточнения в параметрах атрибутов сервиса.	9.1.2

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 8.16

Дата публикации: 28.01.2025.

Изменение	Раздел
Улучшения в версии ТПО 5.164:	
Реализованы новые параметры в элемента **<action>**	4.3, 6.6

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 8.17

Дата публикации: 25.02.2025.

Изменение	Раздел
Улучшения в документации:	
Добавлено описание атрибута **`message`** для тега **<screen>**.	4.1
Добавлено описание флага **`FLAG_VERIFIED_PERSONAL_DATA`** для атрибута платежа.	5.5
Добавлено описание **генерации двумерного штрихкода**.	6.7

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 8.18

Дата публикации: 31.03.2025.

Изменение	Раздел
Улучшения в документации:	
Добавлено описание параметра **trailing-widget** в структуре текстового поля ввода.	5.7

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 8.19

Дата публикации: 05.11.2025.

Изменение	Раздел
Улучшения в документации:	
Добавлено описание флагов	5.5

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 8.20

Дата публикации: 27.01.2026.

Изменение	Раздел
Улучшения в документации:	
Уточнен оператор получения остатка от деления	6.3.3

1 ОСНОВНЫЕ ТЕРМИНЫ

Агент — юридическое лицо, владеющее точками приема платежей или действующее как дистрибьютор, не имея собственных точек приема платежей, и выполняющее функции по приему платежей или предоставлению возможности другим лицам проводить платежи через себя.

Атрибуты платежа — информация, по которой можно идентифицировать операцию платежа в платёжной системе.

Карта сдачи — представляет собой виртуальные денежные средства, которые сохраняются на сервере, при внесении пользователем денежных средств в сумме большей, чем требуется для проведения платежа. Один из возможных вариантов реализации механизма сдачи.

Личный кабинет агента — специализированный веб-сайт, который предназначен для управления сетью точек приема платежей и просмотра финансовой статистики.

Номерная емкость — файл с указанием привязки номера к оператору связи региона.

Объект процессинга — модель данных процессинга содержит в себе описание структур объектов и их отношений. Каждый объект обладает набором определенных свойств и для него могут быть указаны значения этих свойств.

Переменная — поименованная область памяти, адрес которой можно использовать для осуществления доступа к данным. Данные являются значением этой переменной.

Платёж — расчёт за услугу, осуществляемый абонентом (клиентом) при помощи различных платежных инструментов: наличных денежных средств, банковской карты, карт сдачи, ваучеров.

Платежная система — юридическое лицо, самостоятельно предоставляющее возможность оплачивать свои услуги, либо юридическое лицо, предоставляющее возможность оплачивать услуги других компаний.

Тип точки — вид программно-аппаратного устройства, с помощью которого осуществляется прием платежа (терминал, РМА, мобильный клиент и т.д.).

Провайдер платёжных сервисов (от англ. payment service provider) — компания, предоставляющая онлайн-сервисы для проведения электронных платежей различными способами, включая смарт-карты, банковские платежи, такие как банковские операции и др. Некоторые провайдеры платёжных сервисов предоставляют различные инновационные сервисы: платёжные системы, включая платежи наличными, электронные кошельки, предоплаченные карты и т.д.

Проведение платежа — обработка данных о платеже сервером платежной системы.

Произвольное свойство объекта — определяется объекту дополнительно независимо от заданной структуры и взаимосвязей в модели данных. Недоступно в фильтрах, выводится в отчетах по SQL-запросу к БД процессинга.

Сервис — услуга, по которой принимаются платежи в платёжной системе.

Сервис провайдера — сервис в процессинге, жестко привязанный к провайдеру по идентификационному коду в системе провайдера.

Собственный провайдер — провайдер, заведенный на уровне текущего агента, имеющий договор с этим агентом.

Субагент — дочерний агент, находящийся в иерархии агентской сети ниже по отношению к текущему агенту.

Сценарий оплаты по сервису — xml-документ, который описывает процесс оплаты услуги и определяет последовательность полей ввода данных, экранов и правил переходов между ними, допускает как ручной ввода данных, так и получение данных из вне (запросы к шлюзам и оборудованию), при всем этом сценарии поддерживают операции над введенными данными, логические условия, циклы, сложные валидаторы и другие возможности.

Точка — программно-аппаратное устройство, с помощью которого (через которую) осуществляется прием платежа.

Фрейм — область окна браузера для представления отдельной веб-страницы.

2 ПРЕДПОЛАГАЕМАЯ АУДИТОРИЯ

Данное руководство предназначено администраторам и пользователям ПО «Процессинговый центр Pay-logic», имеющим опыт работы с ПС, знакомым с принципами работы в ОС Linux и Windows. Предполагается, что специалист, создающий формы оплаты по сервису обладает общими знаниями в области языков разметки.

Данный документ содержит описание процесса создания и конфигурирования сервисов с использованием сценариев оплаты усовершенствованного модуля ввода данных, а также описание справочной и системной информации ПО «Процессинговый Центр Pay-logic». Язык сценариев, описываемый в руководстве, является разработкой компании Soft-logic и в ПО других производителей не поддерживается.

Примеры, рассматриваемые в документе, применимы в 7 версии ТПО. В тексте документа и в разделе 11 приведены различия синтаксиса 7 и 5 версий ТПО.

Процедуры создания сервисов описаны также в документе «Формы оплаты для универсального модуля ввода данных. Программное обеспечение «Процессинговый центр Pay-logic». Руководство администратора».

3 ОБЩАЯ ИНФОРМАЦИЯ О СЦЕНАРИИ

3.1 ФУНКЦИИ И НАЗНАЧЕНИЕ СЦЕНАРИЕВ

Логика оплаты и последовательность смены экранов переходов на ТПП при оплате услуги описывается по-разному в зависимости от используемого модуля ввода данных.

В ПО «Процессинговый центр Pay-logic» используются три основных модуля ввода данных:

1. Мобильный.

2. Универсальный — подробно описан в документе «Формы оплаты для универсального модуля ввода данных. Программное обеспечение «Процессинговый центр Pay-logic». Руководство администратора».

3. Усовершенствованный.

Основными преимуществами усовершенствованного модуля ввода данных являются:

1. Возможность создания правил перехода на другие экраны (для универсального модуля ввода данных можно задать только последовательность смены экранов).

2. Возможность совершения операций с данными (для универсального модуля ввода данных возможна только проверка данных и форматирование).

Процесс создания сервиса с усовершенствованным модулем ввода данных предполагает выполнение следующих действий:

1. Добавление усовершенствованного модуля ввода данных в процессинг — выполняется администратором ПС.

2. Создание нового сервиса в справочнике сервисов — при создании сервиса в разделе «Справочники — Сервисы — Сервисы» на вкладке «Сценарий» необходимо указать усовершенствованный модуль ввода данных. Подробно параметры,

указываемые при создании сервиса, описаны в документе «Программное обеспечение «Процессинговый центр Pay-logic». Руководство администратора».

3. Связывание сервиса с одним или несколькими провайдерами (определяется направление проведения платежей по сервису).

4. Описание процесса оплаты в файле .xml — сценарии.

5. Создание логотипа.

6. Обновление терминалов и PMA.

Сценарий оплаты по сервису представляет собой .xml-файл определенной структуры. Имя файла сценария совпадает с кодом сервиса. Так, например, сценарий для сервиса с кодом 643 будет иметь название `643.xml`. Как правило, сценарий пишется индивидуально под каждую услугу в связи с особенностями оплаты.

Предусмотрена возможность создания нескольких копий файлов сценариев в зависимости от языка пользователя, то есть поддерживается локализация. Например, у сценария `643.xml` может быть несколько копий, но с текстовками на разных языках (`643.xml`, `643_en.xml`, `643_mk.xml` и т. д.). Язык подгружаемого сценария сопоставляется с выбранным языком пользователя (более подробно локализация описана в разделе 5.3).

Поддерживается возможность двухуровневой загрузки сценария (раздел 5.4).

В ТПО 5 версии сценарии располагаются в каталоге ***<корень ТПО>/resources/scenario/***, в ТПО 7 версии — в каталоге ***<корень ТПО>/res/module/input/advanced/***. Для добавления на терминал новых сценариев, обновления текущих, используется система обновлений, в которой размещаются новые файлы. Настройка системы обновлений описана в документе «Кабинет агента. Программное обеспечение «Процессинговый центр Pay-logic». Руководство пользователя». Для того, чтобы терминал получил обновления, необходимо создать команду на обновление из раздела «Мониторинг — Состояние оборудования» или раздела «Мониторинг — Обновление терминалов» или установить флаг обновления в разделе «Мониторинг — Состояние оборудования» (подробнее в документе «Кабинет агента. Программное обеспечение «Процессинговый центр Pay-logic». Руководство

пользователя»). В системе обновлений сценарии располагаются в каталогах ***/v5/<операционная система>/resources/scenario/*** в 5 версии ТПО и ***/v7/<операционная система>/res/module/input/advanced/*** в 7 версии ТПО, в качестве ОС могут быть указаны linux или windows. Для каждой ОС используется свой каталог в системе обновлений, подробнее в документе «Система обновлений терминального ПО. Программное обеспечение «Процессинговый центр Pay-logic». Руководство пользователя».

В РМА файлы сценариев располагаются в каталоге ***<корень ПО РМА>/resources/scenario/***. В системе обновлений располагаются в каталоге ***<корень ПО РМА>/rma/resources/scenario/***.

Для Andorid-приложений файлы сценариев содержатся в каталоге ***/android/forms/advanced/***.

Кроме того, сценарии могут использоваться в кабинете агента при добавлении платежей через раздел «Диспетчерская — Поиск платежа», «Кассир — Платежи», корректировке платежей, а также валидации введенных параметров. Добавление сценариев в кабинет возможно с использованием двух вариантов:

1. Ручная загрузка в кабинет с использованием раздела «Справочники — Системные параметры — Загрузка файлов» (подробнее в документе «Программное обеспечение «Процессинговый центр Pay-logic». Руководство администратора»).

2. Автоматическая загрузка из системы обновлений. Настройка автоматической загрузки сценариев в кабинет из системы обновлений описана в документе «Система обновлений терминального ПО. Программное обеспечение «Процессинговый центр Pay-logic». Руководство пользователя».

Доступна возможность загрузки в кабинет процессинга сценария по умолчанию с именем `default.xml`. Обработка сценария по умолчанию поддерживается только при создании/корректировке платежей в кабинете. Если сценарий для сервиса не задан, то при создании/корректировке ищется сценарий по умолчанию, если он не найден, то используется форма оплаты номера телефона.

Сценарий описывает экраны ввода данных и экраны выполнения действий (так называемые, void-экраны), а так же логику переходов между ними. Выполнение

сценария начинается со стартового экрана, указанного в атрибуте **`begin`** или **`begin-process`**, если вход в сервис произошел из меню или вследствие перенаправления (редиректа) с другого сервиса, соответственно.

Рассмотрим пример сценария, реализующего оплату кредита.

Сценарий начинается с экрана, на котором пользователь может выбрать один из вариантов оплаты:

1. «Погашение займа клиентом».
2. «Платёж агента по одному договору».
3. «Платёж агента по реестру договоров».

При выборе первого варианта, осуществляется переход на экран, где требуется указать номер договора, а затем, нажав кнопку «Далее», осуществить переход к экрану оплаты, то есть непосредственно к внесению денежных средств на ТПП. Находясь, на экране ввода данных, клиент может вернуться к экрану выбора варианта оплаты, использовав кнопку «Назад», или в главное меню, использовав кнопку «Выход».

При выборе второго варианта, осуществляется переход на экран, где требуется указать номер договора и номер служебного телефона, а затем, нажав кнопку «Далее», осуществить переход к экрану оплаты, то есть непосредственно к внесению денежных средств на ТПП. Находясь, на экране ввода данных, клиент может вернуться к экрану выбора варианта оплаты, использовав кнопку «Назад», или в главное меню, использовав кнопку «Выход».

При выборе третьего варианта, осуществляется переход на экран, где требуется указать номер служебного телефона, а затем, нажав кнопку «Далее», осуществить переход к экрану оплаты, то есть непосредственно к внесению денежных средств на ТПП. Находясь, на экране ввода данных, клиент может вернуться к экрану выбора варианта оплаты, используя кнопку «Назад», или в главное меню, используя кнопку «Выход». Корректность введенных данных при выборе любого варианта оплаты проверяется при помощи совпадения с шаблоном, заданным с помощью регулярных выражений. Алгоритм оплаты, реализуемый сценарием, приведен на рисунке 3.1.1.

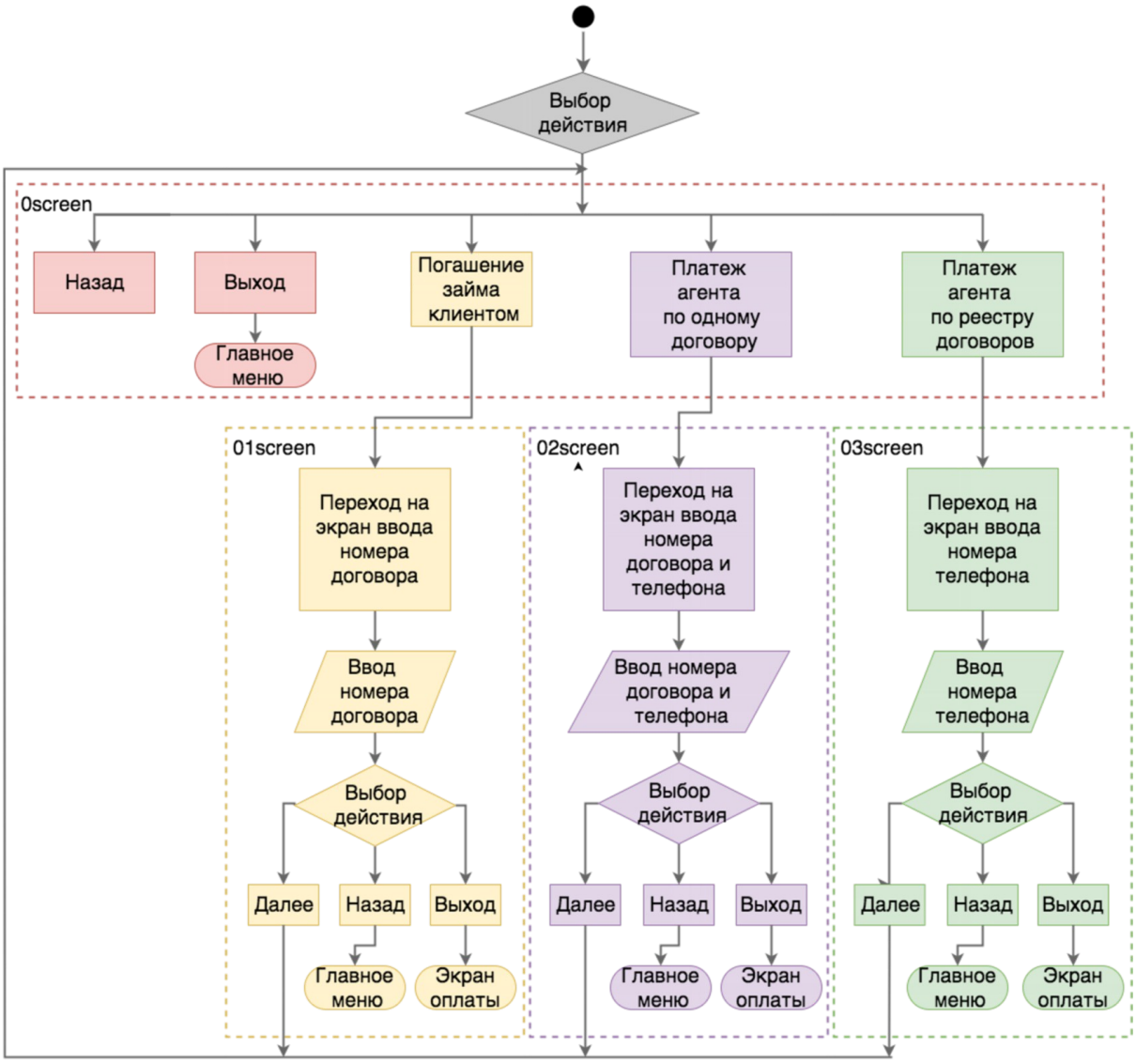


Рисунок 3.1.1 — Алгоритм оплаты, реализуемый сценарием

Код сценария, реализующего данный пример:

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!-- E-Finance  -->
<!--Создание экрана выбора типа оплаты-->
 <screen type="selector" title="Тип оплаты" id="0screen">
  <fields>
   <!--Создание поля выбора типа оплаты-->
   <selector-field id="id2" title="Тип оплаты">
    <items type="static">
     <item value="1" title="Погашение займа клиентом"/>
     <item value="2" title="Платёж агента по одному договору"/>
     <item value="3" title="Платёж агента по реестру договоров"/>
    </items>
   </selector-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="ДАЛЕЕ">
    <!--Если выбран тип оплаты "Погашение займа клиентом", то переход на экран
c id="01screen"-->
    <if condition = "id2 == 1">
     <then> <goto target="01screen"/> </then>
     <else/>
    </if>
    <!--Если выбран тип оплаты "Платёж агента по одному договору", то переход
на экран с id="02screen"-->
    <if condition = "id2 == 2">
     <then> <goto target="02screen"/> </then>
     <else/>
    </if>
<!--Если выбран тип оплаты "Платёж агента по реестру договоров", то переход
на экран с id="03screen"-->
    <if condition = "id2 == 3">
     <then> <goto target="03screen"/> </then>
     <else/>
    </if>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="previous"/>
   </action>
```

```
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="ВЫХОД">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
 <!-- 01screen -->
 <!--Создание экрана для ввода номера договора при выборе типа оплаты
"Погашение займа клиентом"-->
 <screen type="numeric" title="Введите № договора" id="01screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 11. Название поля
ввода: "№ договора"-->
   <text-field id="id1" title="№ договора" max-len="11" keyboard="Digital">
    <validator type="regex">
     <!--Регулярное выражение для валидации вводимого номера-->
     <rules> <rule regex="^\d{11}$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого номера на экране-->
    <formatter type="regex">
     <rules default="**** *****"/>
    </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите № договора </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <goto target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="0screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход"> <goto target="exit"/> </action>
  </actions>
 </screen>
 <!--Создание экрана для ввода номера договора при выборе типа оплаты "Платёж
агента по одному договору"-->
 <!-- 02screen -->
```

```
 <screen type="group" title="Введите № договора,телефона" id="02screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 11. Название поля
ввода: "№ договора"-->
   <text-field id="id1" title="№ договора" max-len="11" keyboard="Digital">
    <validator type="regex">
     <!--Регулярное выражение для валидации вводимого номера-->
     <rules> <rule regex="^\d{11}$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого номера на экране-->
    <formatter type="regex"> <rules default="** *** *** ***"/> </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите № договора </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 10. Название поля
ввода: "№ телефона"-->
   <text-field id="ph" title="№ телефона" max-len="10" keyboard="Digital">
    <validator type="regex">
     <!--Регулярное выражение для валидации вводимого номера-->
     <rules> <rule regex="^\d{10}$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого номера на экране-->
    <formatter type="regex"> <rules default="8(***)***-**-**"/> </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите № служебного телефона </help>
   </text-field>
  </fields>
 <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <set key="sl" value="/" value-title="/"/>
    <!--Склеивание значения id1 "№ договора" и s1 "/"-->
    <str operation="id1 + sl" result-key="id1" result-title="№дог-ра,№тел."/>
    <!--Склеивание значения id1 "№ договора/" и ph "№ телефона"-->
    <str operation="id1 + ph" result-key="id1" result-title="№дог-ра,№тел."/>
    <goto target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад"> <goto target="0screen"/> </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход"> <goto target="exit"/> </action>
```

```
  </actions>
 </screen>
 <!--Создание экрана для ввода номера договора при выборе типа оплаты "Платёж агента по реестру договоров"-→
 <!-- 03screen -->
 <screen type="numeric" title="Введите № телефона" id="03screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая клавиатура, максимальное количество вводимых символов 10. Название поля ввода: "№ телефона"-->
   <text-field id="id1" title="№ телефона" max-len="10" keyboard="Digital">
    <validator type="regex">
     <!--Регулярное выражение для валидации вводимого номера-->
     <rules> <rule regex="^\d{10}$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого номера на экране-->
    <formatter type="regex"> <rules default="8(***)***-**-**"/> </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите № служебного телефона </help>
   </text-field>
  </fields>
 <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее"> <goto target="pay"/> </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад"> <goto target="0screen"/> </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход"> <goto target="exit"/> </action>
  </actions>
 </screen>
</scenario>
```

Стартовый экран определяется идентификатором, указанным в атрибуте **begin** секции:

```
<scenario begin="0screen">
```

Экран «0screen» описывается в секции:

```
<screen type="selector" title="Тип оплаты" id="0screen">
```

Графическое отображение экрана приведено на рисунке 3.1.2.

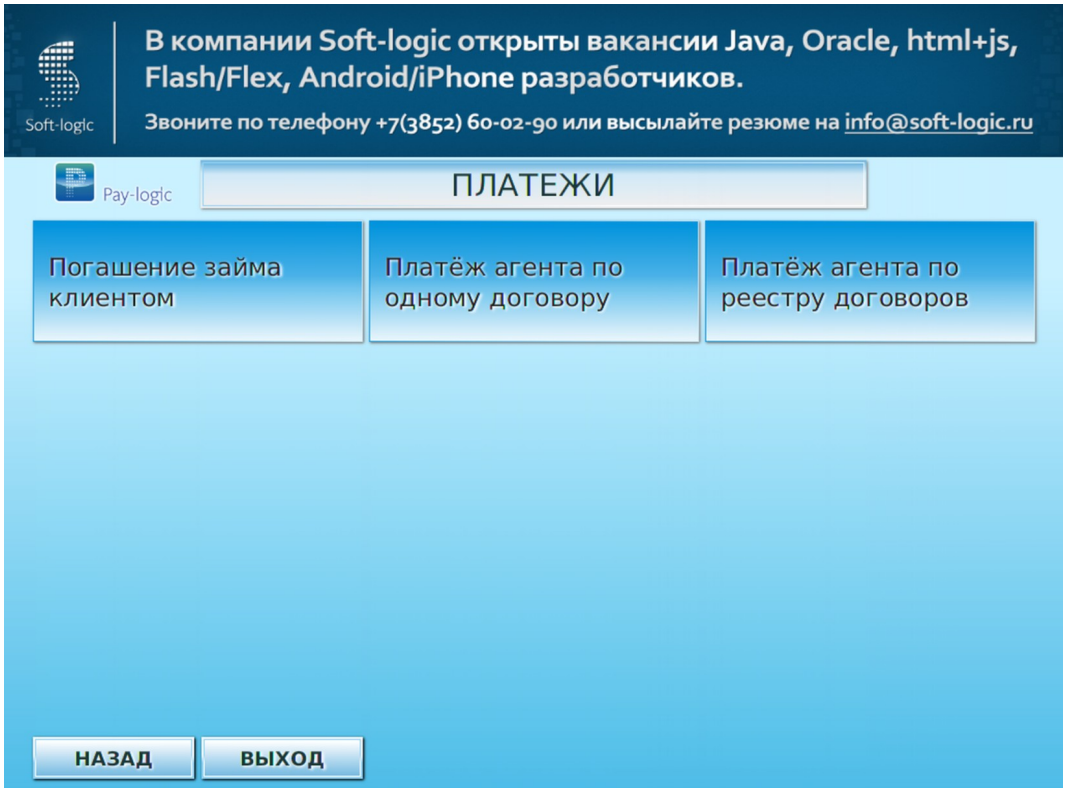


Рисунок 3.1.2 — Стартовый экран «0screen»

Секция `<selector-field id="id2" title="Тип оплаты">` определяет наличие трех вариантов оплаты «Погашение займа клиентом», «Платеж агента по одному договору», «Платеж агента по реестру договоров». Секция `<actions>` определяет, что произойдет при нажатии на любую из кнопок экрана. При выборе варианта оплаты «Погашение займа клиентом» будет осуществлен переход на экран «01screen», при выборе варианта «Платеж агента по одному договору» — «02screen», «Платеж агента по реестру договоров» — «03screen», при нажатии кнопки «Назад» будет осуществлен переход на предыдущий раздел меню, а при выборе кнопки «Выход» будет осуществлен переход в главное меню терминала.

Описание экрана «01screen» начинается с секции:

```
<screen type="numeric"
        title="Введите номер договора"
        id="01screen">
```

Данный экран предусматривает ввод номера договора (рисунок 3.1.3).

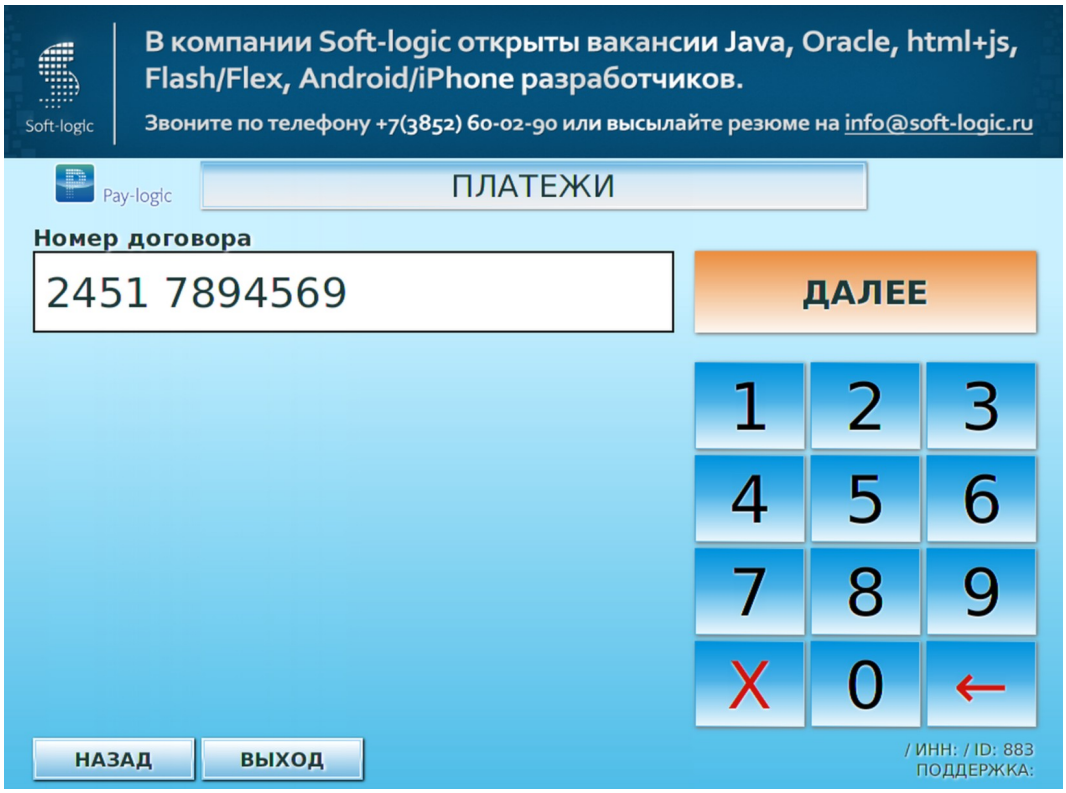


Рисунок 3.1.3 — Экран «01screen»

В секции `<action>` внутри элемента `<screen>`, описывающего данный экран, предусмотрены следующие действия при нажатии кнопок: «Далее» — переход на экран оплаты, «Назад» — переход на предыдущий экран «0screen» (рисунок 3.1.2).

Описание экрана «02screen» начинается с секции:

```
<screen  type="group"
         title="Введите номер договора номер телефона"
         id="02screen">
```

Данный экран предусматривает ввод номера договора и номера телефона (рисунок 3.1.4).

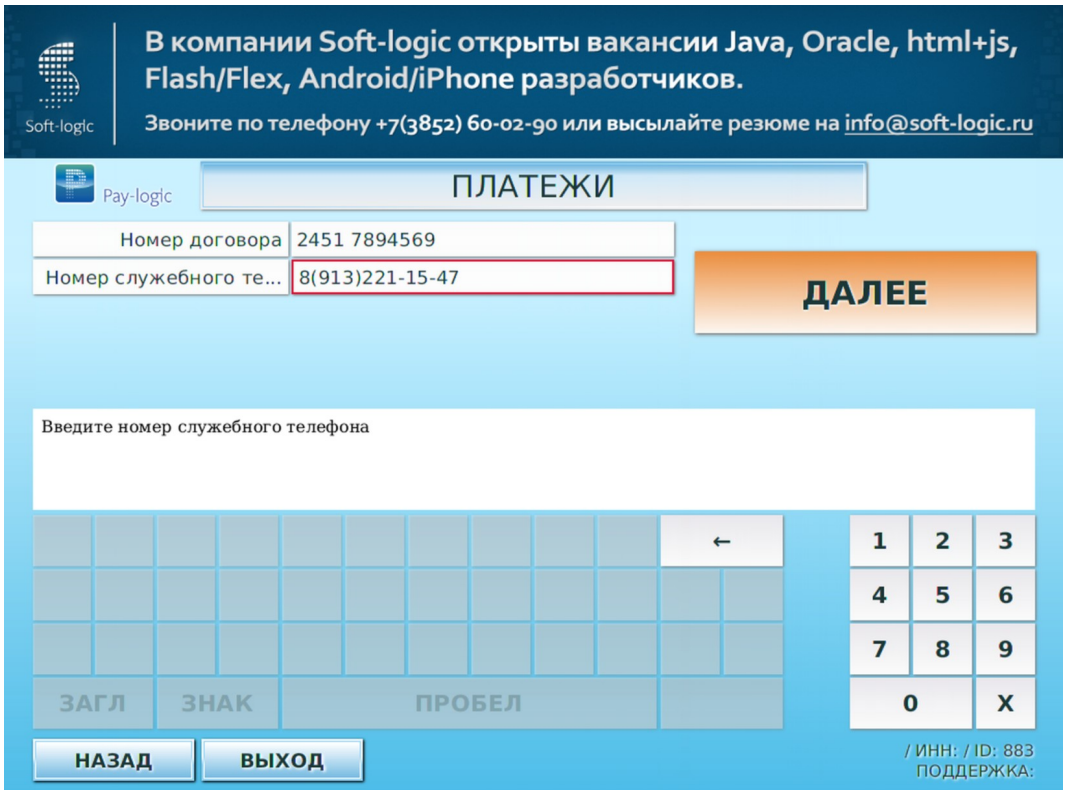


Рисунок 3.1.4 — Экран «02screen»

В секции `<action>` внутри элемента `<screen>`, описывающего данный экран, предусмотрены следующие действия при нажатии кнопок: «Далее» — переход на экран оплаты, «Назад» — переход на предыдущий экран «01screen» (рисунок 3.1.3).

Описание экрана «03screen» начинается с секции:

```
<screen type="numeric"
        title="Введите номер служебного телефона"
        id="03screen">
```

Данный экран предусматривает ввод номера телефона (рисунок 3.1.5).

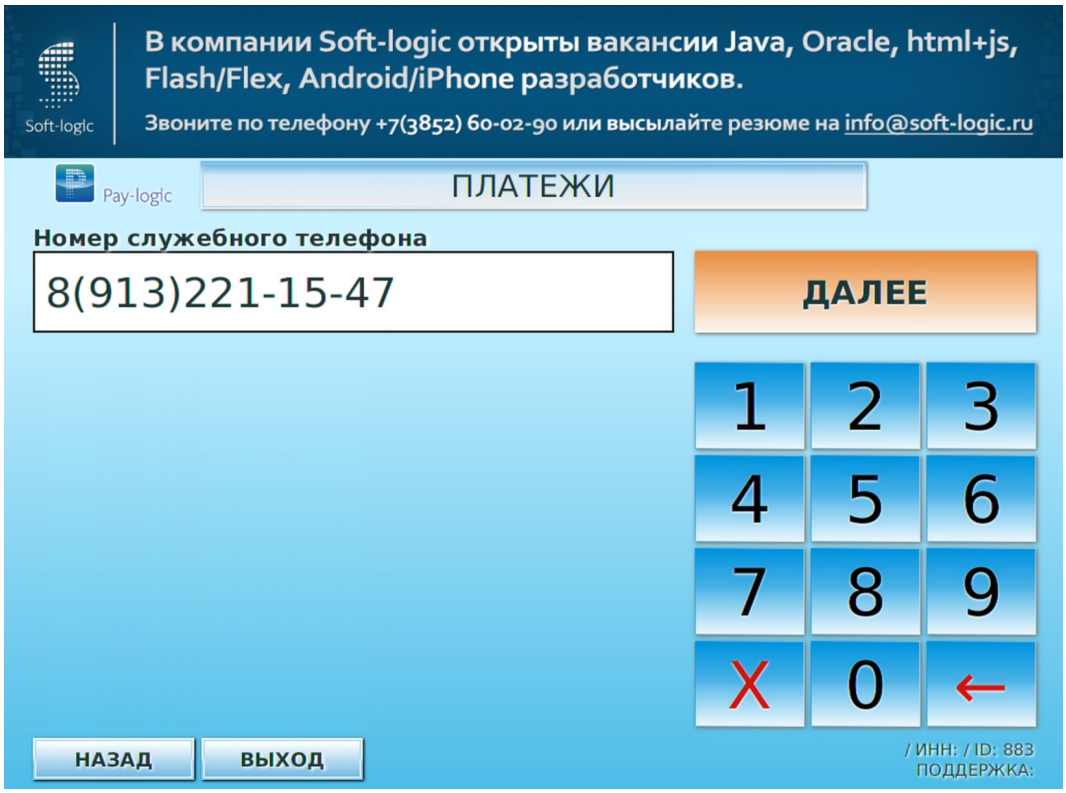


Рисунок 3.1.5 — Экран «03screen»

В секции `<action>` внутри элемента `<screen>`, описывающего данный экран, предусмотрены следующие действия при нажатии кнопок: «Далее» — переход на экран оплаты, «Назад» — переход на предыдущий экран «02screen» (рисунок 3.1.4).

3.2 СТРУКТУРА ФАЙЛА СЦЕНАРИЯ

Файл сценария представляет собой .xml-файл, который должен быть сохранен в кодировке UTF-8. XML-файл хранит информацию о кодировке в XML-декларации, явно указывается следующим образом:

```
<?xml version="1.0" encoding="<кодировка>"?>
<?xml version="1.0" encoding="utf-8"?>
```

Корневой элемент сценария описывается тегом `<scenario>`. Внутри корневого элемента на одном уровне используются экраны ввода, описываемые тегом `<screen>`. Элемент `<screen>` определяет какой вид примет экран ввода и какие функции на этом экране будут доступны. Например, будет ли отображаться на экране одно поле ввода или несколько, будет ли доступна функция считывания штрихкода с документа оплаты, будет ли выполняться проверка введенных данных или выполняться онлайн-запрос при оплате и т. д. Подробное описание экранов приведено в разделе 4. На каждом экране ввода описываются поля ввода данных при помощи элемента `<fields>` — раздел 4.2. В целом, поля ввода данных схожи с полями ввода для универсального модуля ввода данных — «Формы оплаты для универсального модуля ввода данных. Программное обеспечение «Процессинговый центр Pay-logic». Руководство администратора».

Предупреждение!
Объекты сценария (экраны, поля) должны иметь уникальные идентификаторы.

Переходы по экранам и действия с введенными данными по платежу описываются тегом `<actions>` — раздел 4.3, `<navigation>` — раздел 4.4. Сценарии могут обрабатывать события, поступающие от кард-ридера и сканера штрих-кодов — раздел 4.5. В сценариях допускается выносить часть кода в отдельный файл с помощью тега `<xi:include>`, что позволяет не переписывать в каждом сценарии один и тот же фрагмент кода — раздел 6.20.

В тексте XML-документа есть возможность вставлять комментарии. Комментарий начинается с символов "<!--" и заканчивается "-->". С помощью комментариев можно отключать блоки сценария при тестировании, указывать названия экранов перед ними, объяснять действия, происходящие в данном месте сценария и др.

Структура содержимого файла сценария:

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
      <!-- Начальный экран -->
      <screen ...id="0screen">
          <fields> ... </fields>
          <actions> ... </actions>
          <barcode-scanner> ... </barcode-scanner>
          <navigation> ... </navigation>
          <events> ... </events>
          ...
      </screen>
      <!-- Следующий экран -->
      <!-- Включаем файл с ранее описанным типовым экраном -->
      <xi:include .../>
      <!-- Следующий экран -->
      <screen ...>
          <fields> ... </fields>
          <actions> ... </actions>
          <barcode-scanner> ... </barcode-scanner>
          <navigation> ... </navigation>
          <events> ... </events>
          ...
      </screen>
      ...
</scenario>
```

Элемент <scenario> является корневым, с него начинается любой сценарий. Структура:

```
<scenario begin=<screenNameType>
          begin-process=<screenNameType>
          begin-template=<screenNameType>
          begin-template-edit=<screenNameType>
          scard-handler=[bgc|electrical]
          chnum-name=<notEmptyString15>
          features=<decHexInteger>
          check-features=<decHexInteger>>
```

```
        <payment-params> ... <payment-params/>
        <prefill-items> ... <prefill-items/>
        <screen id=<notEmptyString>>
            ...
        </screen>
        ...
</scenario>
```

Дочерние элементы: <payment-params> (раздел 7.2), <prefill-items> (раздел 9), <screen> (раздел 4.1).

Атрибуты описаны в таблице 3.2.1.

Таблица 3.2.1 — Атрибуты элемента <scenario>

Атрибут	Описание	Обяз.
begin	Атрибут, в котором указан идентификатор стартового экрана (экрана, с которого начнется выполнения сценария)	Да
begin-process	Атрибут, в котором передается наименование стартового экрана при перенаправлении из другого сервиса. Если атрибут отсутствует, то в качестве стартового будет использован экран, указанный в атрибуте **begin**	Нет
begin-template	Указывается название стартового экрана для ситуации оплаты шаблона. Если не указано, то считается **null**. Обрабатывается при нажатии кнопки **Next** на экране подтверждения модуля шаблонизации. Если он задан, то переход осуществляется на указанный экран. Если атрибут не задан, то переход осуществляется на экране, указанный в атрибуте **begin**. Работа модуля шаблонизации подробно описана в документе «Кабинет агента. Программное	Нет

Атрибут	Описание	Обяз.
	обеспечение «Процессинговый центр Pay-logic». Руководство пользователя».	
begin-template-edit	Доступен в ТПО 7 на сборках, использующих модуль шаблонизации. Атрибут обрабатывается при нажатии кнопки **Edit** на экране подтверждения модуля шаблонизации. Если он задан, то переход осуществляется на указанный экран. Если атрибут не задан, то переход осуществляется на экране, указанный в атрибуте **begin**	Нет
scard-handler	Указывает какой карточный процессор (модуль работы с смарт-картами) будет использован для данного сценария	Нет
chnum-name	Название атрибута платежа, который будет содержать номер телефона для зачисления сдачи. Значение по умолчанию **«chnum»**	Нет
check-features	При загрузке сценария ТПО автоматически проверяет наличие карточного оборудования (кард-ридер, кард-диспенсер и кард-ридер, поддерживающий возможность работы с чипованными картами), используемого в сценарии и при отсутствии или неисправности одного из устройств отображается сообщение «Использование данной услуги временно невозможно из-за неисправности одного из используемых устройств. Воспользуйтесь сервисом позднее». Атрибут **check-features** позволяет обработать отсутствие/неисправность какого-либо оборудования в сценарии, например, отобразить диалог, указывающий какое именно оборудование недоступно, или осуществить переход к оплате без	Нет

Атрибут	Описание	Обяз.
	использования карточного оборудования. В атрибуте необходимо указать какое оборудование не является обязательным. Возможные значения приведены в таблице 3.2.1.	

Возможные значения **check-features** приведены в таблице 3.2.2.

Таблица 3.2.2 — Значения **check-features**

Использующиеся возможности			Значение check-features
Кард-ридер	**Диспенсер кард**	**Кард-ридер, поддерживающий возможность работы с чипованными картами**	
нет	нет	нет	0
нет	нет	да	1
нет	да	нет	2
нет	да	да	3
да	нет	нет	4
да	нет	да	5
да	да	нет	6
да	да	да	7

Пример (7 версия ТПО):

```
<scenario begin="start" check-features="3">
 <!--Создание экрана выполнения действий-->
 <screen type="void" title="start" id="start">
  <fields/>
  <actions>
```

```
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="Далее">
     <!--Запрос с карт-ридеру-->
     <hdw-request type="card-reader" function="enable" params="$all">
      <actions>
       <!--Описание поведения при успешном запросе к карт-ридеру-->
       <goto-action type="success" target="input_void"/>
       <!--Описание поведения при неуспешном запросе к карт-ридеру-->
       <action type="exception">
        <!--Отображение диалога с сообщением об ошибке-->
        <dialog type="Info" title="Ошибка" message="Ошибка кард-ридера"
                timeout="10" default="okay">
         <actions>
          <!--Описание поведения сценария, если нажата кнопка "OK"-->
          <action type="okay" title="OK">
           <cancel/>
          </action>
         </actions>
        </dialog>
       </action>
      </actions>
     </hdw-request>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <goto-action type="Prev" title="Назад" target="previous"/>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
 ...
</scenario>
```

В данном примере при отсутствии/неисправности кард-ридера будет отображено диалоговое окно «Ошибка кард-ридера». При отсутствии/неисправности диспенсера карт и кард-ридера, поддерживающего возможность работы с чипованными картами, отобразится сообщение «Использование данной услуги временно невозможно из-за неисправности одного из используемых устройств. Воспользуйтесь сервисом позднее».

3.3 ПРОСТЫЕ ТИПЫ ДАННЫХ ЗНАЧЕНИЙ АТРИБУТОВ ЭЛЕМЕНТОВ, ИСПОЛЬЗУЕМЫХ В СЦЕНАРИЯХ

Формат входных данных, обрабатываемых сценариями, описывается input-моделью. Модель содержит описание доступных в сценариях элементов, их атрибутов и типов данных значений атрибутов. В модели используются простые и комплексные типы данных. Комплексные типы содержат набор предопределенных значений. Простые типы, как правило, описывают требования к содержанию значения атрибута. Однако они так же могут содержать предопределенные значения. Комплексные типы данных и простые типы с предопределенными значениями будет описаны при рассмотрении атрибутов с соответствующими типами данных. Ниже приведен список простых типов данных, используемых в input-модели, не имеющих предопределенных значений:

1. **`decHexInteger`** — число в шестнадцатеричной системе счисления.
2. **`dialogMessageType`** — строка длиной до 200 символов.
3. **`fileNameType`** — имя файла. Например, *`resources/ewalletoffer.html`*.
4. **`flags_specificatio`**n — может принимать одно из значений, описанных в разделе 5.5.
5. **`float`** — десятичное число, с двумя цифрами в дробной части. Минимальное значение «0.00».
6. **`integer`** — целое число.
7. **`keyboard_specification`** — значение указывается в соответствии с правилами, описанными в разделе 5.6.
8. **`keyType`** — строка, первый символ обязательно буква, последующие: буквы, цифры, "-","_", длина от 1 до 25 символов.

9. **keyListType** — строка первый символ может быть #, далее любая буква латиницы, за ней символы латиницы, цифры, символы «.», «-», «_» в количестве от 1 до 30. Этой группы символов может не быть. Далее следует любое количество пробельных символов, запятая, любое количество пробельных символов, далее название переменной как описано выше и эта строка может повторяться любое количество раз. Вместо данной записи может быть указано «$all».

10. **messageType** — непустая строка от 1 до 40 символов. Первый символ — прописная буква, или цифра, или символ «№», следующие: любые. Последний: обязательно непробельный символ.

11. **nonNegativeInteger** — положительное целое число и 0.

12. **notEmptyString(n)** — текстовая строка длиной до n символов. Если n не указано, то ограничение на максимальную длину отсутствует.

13. **positiveFloat** — положительное вещественное число, минимальное значение «0».

14. **positiveInteger** — положительное целое число.

15. **positiveShort** — положительное целое число в шестнадцатеричной системе счисления.

16. **resourceIdType** — идентификатор. Идентифицирует сущность, определенную в другом месте. Представлен строкой, содержащей от 2 до 30 символов, первый символ обязательно строчная буква латинского алфавита, последующие символы: цифры от 0 до 9, строчные буквы латинского алфавита и символы «.», «_», «-».

17. **selectorTitleType** — непустая строка от 1 до 35 символов. Первый символ: прописная буква, или цифра, или символ «№», следующие: любые за исключением пробелов, табуляций и переносов строк. Последний: обязательно непробельный символ.

18. **screenNameType** — непустая строка от 1 до 35 символов. Может содержать цифры, прописные и строчные буквы, символы «\», «-», «_».

19. **srvId** — идентификатор сервиса, целое положительное число, минимальное значение: «0», максимальное: «32767».

20. **string** — строка без ограничений на длину и символы, в нее входящие.

21. **tId** — строка, состоящая из букв заглавных, прописных, любое количество повторений.

22. **titleType** — непустая строка от 1 до 30 символов. Первый символ: прописная буква, или цифра, или символ «№», следующие: любые. Последний: обязательно непробельный символ.

4 ЭКРАНЫ

4.1 ЭКРАН (<SCREEN>)

Элемент `<screen>` описывает один экран сценария, экраны отличаются по типу, по оформлению. В зависимости от реализации пользовательского интерфейса, могут не работать некоторые значения атрибутов тега `<screen>`. Структура элемента:

```
<screen type=[barcode|bustickets|cash|communal|communalcounter|
              communal/meter|confirm|confirmbig|confirm/commission|
              confirm/long|confirm/navi|confirm/multiple|
              confirm/multiple/navi|date|date/popup|digital|
              digital/commission|digital/info|group|group/chcopy|
              group/meter|group/navi|group/numeric|group/numeric/navi|
              group/popup|info|info/agree|info/comm|info/navi|letter|
              numeric|numeric/cbc|numeric/navi|numeric/popup|selector|
              selector/button|selector/cards|selector/image|
              selector/navi|selector/variant|sum|table|table/navi|void|
              communal/charge/uni|confirm/communal]
         id=<notEmptyString>
         title=<titleType>
         title-id=<resourceIdType>
         message=<messageType>
         decor=[banner|commission|long|info|button|cards|variant]
         focused-field=<string>>
</screen>
```

Дочерние элементы для всех типов экрана: `<fields>`, `<actions>`, `<goto-action>`, `<navigation>`, `<events>`, `<barcode-scanner>`.

Атрибуты для всех типов экранов описаны в таблице 4.1.1.

Таблица 4.1.1 — Атрибуты для всех типов экранов

Атрибут	Описание	Обяз.
type	Определяет функциональный тип экрана — основную	Да

Атрибут	Описание	Обяз.
	функцию, которую он выполняет. Тип экрана нужно указывать со строчной буквы в обеих версиях ТПО. Через слеш «/» указывается оформление экрана (его внешний вид, декор). **Пример:** `type=«confirm/navi»` Ранее оформление указывалось в специальном атрибуте **`decor`**. **Обратите внимание**, что только для экрана оплаты оформление может задаваться как в сценарии, так и в кабинете при редактировании сервиса в профиле меню. Декор, указанный в сервисе, затем передается в составе справочников на точку. Приоритет декора из сценария выше, чем значение декора из сервиса (справочника). При этом: • Если в сценарии указан тип экрана **sum**, а декор не указан, ТПО посчитает, что значение не задано, и использует настройки из справочника. Однако если в справочнике тоже нет информации, ТПО вновь обратится к сценарию и будет использовать тип экрана **sum** без декора	
`id`	Указывается ключ атрибута платежа	Да
`title`	Отображаемое название экрана	Да
`title-id`	Содержит идентификатор текстовки, которая будет подгружена в качестве значения **`title`**	Нет
`message`	Содержит текстовое сообщение к экрану	Нет
`sound`	Название звукового файл, который необходимо воспроизвести при переходе на экран	Нет
`focused-field`	Определяет поле, в которое переместится фокус	Нет

Пример использования `focused-field` (фокус переместится в поле атрибута **`counter8_tp`**):

```
<scenario xmlns="http://pay-logic.ru" begin="counter8">
<!--Создание экрана для ввода данных по счётчику,при открытии экрана курсор
установлен в поле id="counter8_tp" "Текущие показания счетчика 8"-->
<screen type="group" title="Введите данные по счетчику 8" id="counter8"
        focused-field="counter8_tp">
 <fields>
 <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура с возможностью переключения языка,
максимальное количество вводимых символов 60. Название поля ввода: "Код
счетчика 8"-->
 <text-field id="counter8" title="Код счетчика 8" max-len="60"
             keyboard="any:[ru,en,symb]:lower:true" read-only="true">
  <!--Регулярное выражение для валидации вводимого значения-->
  <validator type="regex">
   <rules>
    <rule regex="^.{1,60}$" />
   </rules>
  </validator>
 </text-field>
 <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура с возможностью переключения языка,
максимальное количество вводимых символов 60. Название поля ввода:
"Наименование счетчика 8"-->
 <text-field id="counter8_name" title="Наименование счетчика 8" max-len="60"
             keyboard="any:[ru,en,symb]:lower:true" read-only="true">
  <!--Регулярное выражение для валидации вводимого значения-->
  <validator type="regex">
   <rules>
    <rule regex="^.{1,60}$" />
   </rules>
  </validator>
 </text-field>
 <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура с возможностью переключения языка,
максимальное количество вводимых символов 60. Название поля ввода:
"Предыдущие показания счетчика 8". Поле неактивно для ввода-->
 <text-field id="counter8_pp" title="Предыдущие показания счетчика 8"
             max-len="60" keyboard="any:[ru,en,symb]:lower:true"
             read-only="true">
  <!--Регулярное выражение для валидации вводимого значения-->
```

```
 <validator type="regex">
  <rules>
   <rule regex="^.{1,60}$" />
  </rules>
 </validator>
</text-field>
<!--Создание поля ввода числа в десятичном формате через точку. Название поля
ввода: "Текущие показания счетчика 8"-->
 <numeric-field id="counter8_tp" title="Текущие показания счетчика 8"
                format="6.3">
  <!--Подсказка, отображается на экране-->
  <help> Введите текущие показания счетчика 8 </help>
   <!--Регулярное выражение для валидации вводимого значения-->
   <verify>
    <range begin="0.00" end="999999.999"/>
   </verify>
 </numeric-field>
</fields>
<actions>
 ...
</actions>
</screen>
</scenario>
```

Доступные виды экранов приведены в таблице 4.1.2.

Таблица 4.1.2 — Доступные виды экранов

Тип экрана	Описание	Поддержка в ТПО 5, интерфейс BlueSphere2	Поддержка в ТПО 5, интерфейс Smoke	Под-держка в ТПО 7
BARCODE	Экран для ввода штрих-кода, расшифровывает полученную строку	-	-	-
BUSTICKETS	Экран выбора мест в автобусе	+	-	-
COMMUNAL	Экран для выбора	+	+	+

Тип экрана	Описание	Поддержка в ТПО 5, интерфейс BlueSphere2	Поддержка в ТПО 5, интерфейс Smoke	Под-держка в ТПО 7
	нескольких услуг и общей оплаты по ним			
COMMUNAL/BASKET	Экран корзины платежей при групповом платеже по разным сервисам	-	+	-
COMMUNAL/CHARGE/UNI	Коммунальный экран	-	-	+
COMMUNAL/METER	Экран для ввода показаний счетчиков и приборов учета	-	-	+
CONFIRM	Экран для подтверждения введенной ранее информации (может отображать до 8 полей)	+	+	+
CONFIRM-BIG	Экран для подтверждения введенной ранее информации (может отображать до 3-х полей)	+	-	+
CONFIRM/COMMISSION	Экран подтверждения введенной информации с	+	-	-

Тип экрана	Описание	Поддержка в ТПО 5, интерфейс BlueSphere2	Поддержка в ТПО 5, интерфейс Smoke	Под-держка в ТПО 7
	отображением информации о комиссии по сервису			
CONFIRM/COMMUNAL	Экран подтверждения в виде табличного счета	-	-	+
CONFIRM/LONG	Представляет собой незначительно измененный экран подтверждения	+	-	-
CONFIRM/NAVI	Экран подтверждения с навигацией	-	-	+
CONFIRM/MULTIPLE	Экран подтверждения с несколькими полями	-	-	+
CONFIRM/MULTIPLE/NAVI	Экран подтверждения с несколькими полями и навигацией	-	-	+
DATE	Экран для выбора даты	-	+	+
DATE/POPUP	Всплывающий экран выбора даты	-	-	+
DIGITAL	Экран для ввода числовой и текстовой	+	-	+

Тип экрана	Описание	Поддержка в ТПО 5, интерфейс BlueSphere2	Поддержка в ТПО 5, интерфейс Smoke	Под-держка в ТПО 7
	информации			
DIGITAL/COMMISSION	Экран для ввода числовой и текстовой информации с отображением информации о комиссии	+	-	-
DIGTAL/INFO	Экран для ввода числовой и текстовой информации с отображение подсказки. См. раздел 10.2	+	-	-
GROUP	Экран вывода нескольких полей ввода различных типов	-	+	+
GROUP/CHCOPY	Экран вывода нескольких полей ввода на экране печати копии чека	-	-	+
GROUP/CHECKBOX	Групповой экран с чек-боксом	-	+	-
GROUP/METER	Экран для ввода текущих показаний счетчиков и суммы	-	-	+

Тип экрана	Описание	Поддержка в ТПО 5, интерфейс BlueSphere2	Поддержка в ТПО 5, интерфейс Smoke	Под-держка в ТПО 7
	оплаты			
GROUP/NAVI	Экран для вывода нескольких полей ввода с навигацией	-	-	+
GROUP/NUMERIC	Экран для вывода нескольких числовых полей ввода	-	+	+
GROUP/NUMERIC/NAVI	Экран для вывода нескольких числовых полей ввода с навигацией	-	-	+
GROUP/POPUP	Всплывающий экран для вывода нескольких полей ввода различных типов	-	-	+
INFO	Экран для вывода вспомогательной информации: помощь, предупреждение, выдержки из законодательных актов и т.д.	+	+	+
INFO/AGREE	Экран для вывода плательщику информации об	-	-	+

Тип экрана	Описание	Поддержка в ТПО 5, интерфейс BlueSphere2	Поддержка в ТПО 5, интерфейс Smoke	Под-держка в ТПО 7
	условиях оплаты, с которыми он может согласиться или не согласиться			
INFO/COMM	Экран для вывода плательщику информации о действующей комиссии	-	-	+
INFO/NAVI	Экран для вывода плательщику информации с элементами навигации	-	-	+
NUMERIC	Экран для ввода числовой и текстовой информации	-	+	+
NUMERIC/CBC	Экран для ввода пин-кода карты сдачи	-	-	+
NUMERIC/NAVI	Экран для ввода числовых данных с навигацией	-	-	+
NUMERIC/POPUP	Всплывающий экран ввода числовых данных	-	-	+
SELECTOR	Экран для вывода информации с	+	+	+

Тип экрана	Описание	Поддержка в ТПО 5, интерфейс BlueSphere2	Поддержка в ТПО 5, интерфейс Smoke	Под-держка в ТПО 7
	последующим выбором одного из предложенных пунктов			
SELECTOR/BIGBUTTON	Экран для вывода информации в виде больших кнопок	-	+	-
SELECTOR/BUTTON	Экран выбора	+	+	-
SELECTOR/CARDS	Экран для вывода информации в виде кнопок, на которых отображаются изображения, с возможностью последующего выбора одного из предложенных пунктов (графический селектор)	+	-	+
SELECTOR/IMAGE	Графический селектор, указание атрибута **sum** необязательно	+	+	+
SELECTOR/NAVI	Экран вывода нескольких вариантов с навигацией	-	-	+

Тип экрана	Описание	Поддержка в ТПО 5, интерфейс BlueSphere2	Поддержка в ТПО 5, интерфейс Smoke	Под-держка в ТПО 7
SELECTOR/VARIANT	Возможно размещение четырех кнопок и полей информации справа	+	-	-
SUM	Экран для переопределения атрибута, отображаемого на экране оплаты	+	+	+
SUM/CASHLESS	Экран оплаты при оплате картой	+	+	-
TABLE	Экран для вывода данных в виде таблицы	-	+	+
TABLE/NAVI	Экран для вывода таблицы с навигацией по таблице на каждом экране	-	-	+
VOID	Экран для выполнения каких-либо действий	+	+	+
CinemaTickets	Экран для выбора мест в кинотеатре	+	-	-

Внешний вид экрана ТПО определяется значением атрибута **`type`**, в качестве которого могут быть указаны варианты, записанные в таблице 4.1.2 в столбце «Тип экрана», в случае, если в ячейке на пересечении строки с типом экрана и столбца поддержки в нужной версии ТПО установлен символ «+».

Значения атрибутов **`title`** и **`id`** задаются самостоятельно по смыслу.

4.2 ПОЛЯ (<FIELDS>)

4.2.1 ОБЩИЕ ПОЛОЖЕНИЯ

В секции `<fields>` описываются список полей ввода, доступных на экране. Не используется на экране «VOID». Поле ввода определяет, какие данные пользователь может указать в качестве значения. Существуют следующие поля ввода:

1. Поле ввода текста — `<text-field>`.
2. Числовое поле ввода — `<numeric-field>`.
3. Флаг/переключатель — элемент, позволяющий управлять параметром с двумя состояниями(вкл/выкл), `<checkbox-field>`.
4. Селекторы различных разновидностей (выбор варианта из предложенных списков, в том числе динамических) — `<selector-field>`, `<table-field>`.
5. Поля для ввода даты — `<date-field>`.
6. Поле с автозаполнением — `<autocomplete-field>`.
7. Поле расшифровки информации, считанной сканером штрих-кодов— `<barcode-field>`, используется только в 5 версии ТПО.

В полях ввода данных осуществляется инициализация переменных, то есть атрибутов платежа. Названия переменных произвольные, по смыслу, например, если поле ввода предполагает указание названия улицы или номера дома, то правильно их стоило бы назвать **`street`** и **`house`** соответственно. Наименование переменной задается в атрибуте **`id`** поля ввода (раздел 5.2). В усовершенствованном типе обработчика введены предопределенные переменные **`id1`** и **`id2`**. Их длина не должна превышать 40 символов. Как правило, эти переменные служат для передачи номера телефона,

лицевого счета или иной уникальной информации. При работе с универсальным обработчиком в форме оплаты обязательно должна присутствовать переменная **id1**, при работе с усовершенствованным обработчиком можно не включать переменную **id1** в сценарий в случае, если используются оффлайн-платежи. Однако рекомендуется в таком случае все-таки определить **id1** пустой строкой.

В зависимости от интерфейса внутри `<fields>` возможно использовать элемент `<row>`, позволяющий разместить указанные внутри него поля ввода в одну строчку. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper».

4.2.2 ОБЪЕДИНЕНИЕ ПОЛЕЙ В СТРОКУ (<ROW>)

Элемент `<row>` позволяет разместить указанные внутри него поля ввода в одну строчку. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper».

Синтаксис:

```
<row orientation=[PORTRAIT|LANDSCAPE]>
      <text-field width=<nonNegativeInteger>
          ...
      </text-field>
      ...
      <placeholder width=<nonNegativeInteger>/>
      ...
</row>
```

Значения PORTRAIT и LANDSCAPE задают для какой ориентации используется размещение в одну строку: PORTRAIT — вертикальная, LANDSCAPE — горизонтальная.

При таком варианте использования для всех полей модели доступны атрибуты **width**, **rowId**.

Атрибут **width** определяет ширину поля, для которого он указан, в текущей строке в процентах. Если сумма значений **width** всех полей внутри `<row>` превышает 100, то

возникнет ошибка при обработке сценария. Если сумма значений **width** всех полей внутри `<row>` превышает 100, но **width** указан не для всех полей, то возникнет ошибка при обработке сценария. Если значение не указано, то считается, что такое поле занимает все свободное место. Если таких полей несколько, то они делят все свободное место поровну.

Атрибут **rowId** определяет идентификатор строки, в которой будет расположено поле.

Элемент `<placeholder>` является полем ввода, но не несет никакой полезной нагрузки кроме значения атрибута **width**.

4.3 ДЕЙСТВИЯ (<ACTIONS>)

Элемент `<actions>` описывает список кнопок, расположенных на экране, и действия, соответствующие этим кнопкам. Структура элемента:

```
<actions>
          <action> ... </action>
          ...
</actions>
```

Дочерние элементы: `<action>`, `<goto-action>` — взаимозаменяющие элементы.

Атрибуты: отсутствуют.

Элемент, задаваемый тегом `<action>`, описывает реакцию на нажатие одной из кнопок. В зависимости от реализации пользовательского интерфейса возможно использование на одном экране до четырех элементов `<action>`. Содержание действия задается последовательностью команд внутри элемента `<action>`, которые выполняются последовательно, если иное не задано самими командами. Например, на экране ввода штрих-кода кнопки **«Пропустить»**, **«Далее»**, **«Назад»**, **«Отмена»**. Структура:

```
<action type=[Next|Prev|Skip|Edit|Exit|SelectALL|RemoveAll|screenaction]
        title=<titleType>
        title-id=<resourceIdType>>
           ...
</action>
```

Элемент `<action>` с типом **«`Next`»** предполагает переход к следующему экрану.

Примечание!
Действие «Пропустить» (Skip) доступно только на экране «BARCODE». Действия «Выбрать все» (SelectALL), «Удалить все» (RemoveAll) доступны только при работе с корзиной.

Например, переход к экрану «1screen», следующему за текущим, осуществляется с помощью конструкции:

```
<action type="Next" title="ДАЛЕЕ">
      <goto target="1screen"/>
</action>
```

Переход к экрану оплаты осуществляется с помощью конструкции:

```
<action type="Next" title="ДАЛЕЕ">
      <goto target="pay"/>
</action>
```

Доступна возможность перехода к экрану посредством указания атрибута, содержащего название экрана, на который необходимо перейти. Таким образом, переход к экрану «1screen» может быть выполнен с помощью конструкции:

```
<action type="Next" title="ДАЛЕЕ">
      <goto target="${nextScreen}"/>
</action>
```

где атрибут **nextScreen** будет иметь значение, соответствующее названию экрана, на который выполняется переход:

```
nextScreen="1screen"
```

Элемент `<action>` с типом **«Prev»** предполагает переход к предыдущему экрану. Например, переход к предыдущему экрану «1screen» осуществляется с помощью конструкции:

```
<action type="Prev" title="Назад">
      <goto target="1screen"/>
</action>
```

Выход в меню по нажатии кнопки «Назад» осуществляется с помощью конструкции:

```
<action type="Prev" title="Назад">
  <goto target="previous"/>
</action>
```

Элемент `<action>` с типом **«Exit»** предполагает выход в главное меню:

```
<action type="Exit" title="Выход">
      <goto target="exit"/>
</action>
```

Принципиальным отличием конструкций является то, что **«Exit»** перенаправляет пользователя на домашний экран приложения (главное меню), а **«Previous»**

возвращает пользователя на тот уровень меню, где он находился до выбора сервиса. Предпочтительно на всех первых экранах использовать **«Previous»**, а на последующих — **«Exit»**. Это позволит пользователю вернуться сразу к выбору сервиса (избежав необходимости повторного перехода по всей структуре меню) по нажатию кнопки «Назад», в случае, если первоначально он промахнулся и выбрал не тот сервис.

Элемент `<action>` с типом **«Edit»** позволяет выполнять любые действия из таблицы 4.3.1:

```
<action type="edit" title="">
      <set key="tariff" value="2.5"/>
      <goto target="input"/>
</action>
```

Внимание!
Отображение кнопки **«Edit»** на экранах требует доработки интерфейса. В 7 версии ТПО обработка кнопки **«Edit»** поддерживается на экранах **«confirm»** (раздел 4.6.6), **«group»** (раздел 4.6.18), **«info»** (раздел 4.6.23), **«numeric»** (раздел 4.6.28).

Также обрабатывается элемент `<action>` с типом **«screenaction»**. Элемент `<action>` с типом **«screenaction»** предполагает выполнение команд в момент перехода с другого экрана и отрисовки текущего экрана. В момент выполнения обработки действия другие кнопки на экране недоступны, но, как правило, неактивность кнопок плательщику незаметна, так как команды обрабатываются быстро. В том случае, если в секции **«screenaction»** не было перехода на другой экран, отмены или выхода из сценария, текущий экран будет перерисован. Таким образом, на нем будут доступны установленные значения переменных.

Пример:

```
<!--Создание информационного экрана-->
<screen id="show" type="info" title="Информация">
 <field/>
```

```
 <actions>
  <!--Описание действий, выполняемых в момент отрисовки текущего экрана-->
  <action type="screenaction" title="Default">
   <!--Объявление переменной id1-->
   <set key="id1" key-title="id1" value="0" value-title="0" />
   <!--Объявление переменной #photo-number-->
   <set key="#photo-number" key-title="Photo #2" value="2" value-title="2" />
   <!--Включение в сценарий данных из внешнего файла-->
   <xi:include href="res/module/input/advanced/storeimage.xml" />
  </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
  <action type="Next" title="ДАЛЕЕ">
   <goto target="pay"/>
  </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
  <action type="Prev" title="Назад">
   <goto target="previous"/>
  </action>
 </actions>
</screen>
```

Перечень доступных команд приведен в таблице 4.3.1.

Таблица 4.3.1 — Перечень доступных команд

Описание	Элемент
Отмена действия	<cancel>
Удаление всех ранее инициализированных переменных	<clear-all>
Удаление переменных из контекста	<clear>
Удаление значений всех инициализированных в сценарии переменных за исключением, указанных в атрибутах элемента	<clear-except>
Удаление переменных из контекста, соответствующих регулярному выражению	<clear-like>
Сложная валидация данных	<complex-validator>
Осуществление математических операций	<math>
Преобразование строковых переменных	<modify>

Описание	Элемент
Конвертация значений атрибутов	<convert>
Форматирование даты	<date-util>
Замена символов	<encode>
Объявление переменной	<set>
Конкатенация строковых переменных	<str>
Создание атрибута для отображения на экране подтверждения	<text-format-table>
Циклический оператор	<for> (доступен только в 7 и 5 версии ТПО)
Операторы условного перехода	<if>, <switch>
Оператор безусловного перехода	<goto>
Онлайн-запрос	<online-request>
Запрос к оборудованию	<hdw-request>
Перенаправление	<redirect>
Интерактивные диалоги	<dialog>
Указание ограничения максимальной и минимальной зачисленных сумм	<set-sum-limit>
Вычисление суммы платежа без комиссии	<set-sum>
Вычисление итоговой суммы	<sum>
Подготовка атрибутов платежа	<communal-prepare>
Указание особенностей проведения группового платежа	<payment-params>
Упаковка элементов платежа	<pack>
Распаковка элементов платежа	<unpack>
Предзаполнение полей данными из окружения	<load>
Включение данных из внешнего файла	<xi:include>
Печать чеков без оплаты	<print>
Генерация уникальных последовательностей	<sequence-generator>

Описание	Элемент
Открытие html-страниц	<open-url>
Получение данных с карточного модуля (доступен в версии ТПО 5, в 7 версии — недоступен)	<card-module-task>
Функция расчета хеша	<hash>
Вызов внешнего скрипта	<execute-script>
Чтение треков карты	<read-card-track>

Атрибуты элемента <action>: type, title

1. **type** — тип кнопки («**Next**» — далее, «**Prev**» — назад, «**Skip**» — пропустить, «**Exit**» — выход, «**Edit**» — редактировать, «**SelectALL**» — выбрать все, «**RemoveAll**» — удалить все, «**screenaction**» — действие по умолчанию).

2. **title** — отображаемая надпись на кнопке.

3. **title-id** — содержит идентификатор текстовки, которая будет подгружена в качестве значения **title**.

4. **execute-after-timeout** – в атрибуте задается время в секундах (от 1 до 120), после достижения которого будет выполнено действие, заданное в **<action>.** Обрабатывается ТПО 5.

В зависимости от содержания информации и необходимости выполнения действий с введенными данными по платежу можно вырезать подстроки, проводить простейшие математические операции, стирать информацию, осуществлять переход к заданному экрану и пр.

Пример:

```
<screen ...      >
   <actions>
      <action type="Next" title="ДАЛЕЕ">
         <goto target="id_следующего_экрана"/>
      </action>
```

```
    <action type="Prev" title="НАЗАД">
       <goto target="id_предыдущего_экрана"/>
    </action>
  </actions>
</screen>
```

Если других вложенных действий в <action>, кроме <goto> не предполагается, то целесообразно использование <goto-action>.

Дочерние элементы: отсутствуют.

Атрибуты элемента <goto-action> приведены в таблице 4.3.2.

Таблица 4.3.2 — Атрибуты элемента <goto-action>

Атрибут	Описание	Обяз.
type	Идентификатор <goto-action>	Да
title	Заголовок элемента навигации	Нет
target	Идентификатор экрана, на который осуществляется переход	Да

Пример:

Вместо конструкции:

```
<action type="navi1" title="Данные платежа">
      <goto target="1screen"/>
</action>
```

возможно использовать конструкцию:

```
<goto-action type="navi1" title="Данные платежа" target="1screen"/>
```

4.4 НАВИГАЦИЯ (<NAVIGATION>)

Для языка сценариев в 7 версии ТПО доступен тип экранов с навигацией. Экраны данного типа могут отображать навигационные кнопки, которые соответствуют шагам сценария или формы. При помощи данных кнопок можно осуществить возврат назад на любой пройденный этап сценария или формы. Экраны с навигацией имеют такой же тип как и обычные, только с постфиксом **«/navi»**. Описание навигации осуществляется в секции `<navigation></navigation>`. В секции `<action>` описывается переход на конкретный шаг (экран) сценария.

Дочерние элементы: `<action>`.

Атрибуты: отсутствуют.

Структура элемента:

```
<action type=<string>
        title=<titleType>
        title-id=<resourceIdType>
        current=[true|false]>
      <goto target=<screenNameType>/>
</action>
```

Атрибуты элемента **<action>** приведены в таблице 4.4.1.

Таблица 4.4.1 — Атрибуты элемента **<action>**

Атрибут	Описание	Обяз.
type	Идентификатор `<action>`	Да
title	Заголовок элемента навигации	Нет
title-id	Содержит идентификатор текстовки, которая будет	Нет

Атрибут	Описание	Обяз.
	подгружена в качестве значения **`title`**	
`current`	Флаг, показывающий является ли данный `<action>` (шаг) текущим. Возможные значения: • **`true`** — является текущим; • **`false`** — не является текущим	Нет

Порядок следования `<action>` определяет порядок следования кнопок навигации на экране. Кнопки шагов после текущего неактивны, то есть переход возможен только на предыдущие шаги. Элемент `<action>` может содержать в себе любые действия доступные данному полю, например онлайн-запросы, переходы на экраны и т.д.

Полный текст сценария с навигацией:

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!--Создание экрана ввода номера брони-->
 <screen type="numeric" title="Введите номер брони" id="0screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 6. Название поля ввода:
"Номер брони"-->
   <text-field id="id1" title="Номер брони" max-len="6" keyboard="Digital">
   <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules> <rule regex="^.{6}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите номер брони]]> </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <goto-action type="Next" title="Далее" target="1screen"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <goto-action type="Prev" title="Назад" target="previous"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
```

```
 </screen>
 <!--Создание экрана ввода данных плательщика с несколькими полями ввода-->
 <screen type="group" title ="Введите данные" id="1screen">
   <fields>
    <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский, без возможности
смены, максимальное количество вводимых символов 60. Название поля ввода:
"ФИО плательщика"-->
    <text-field id="fio" title="ФИО плательщика" max-len="60"
                keyboard="any:[ru,symb]:upper:true">
    <!--Регулярное выражение для валидации вводимых данных-->
     <validator type="regex">
       <rules>
        <rule regex="^[a-яА-Яёё-]{2,20}[\s]{1}[a-яА-Яёё-]{2,20}[\s]{1,3}
                    [a-яА-ЯЁё-]{2,20}([\s]{1}[a-яА-ЯЁё-]{2,20})?$"/>
       </rules>
     </validator>
     <!--Подсказка, отображается на экране-->
     <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
    </text-field>
    <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский, без возможности
смены, максимальное количество вводимых символов 254. Название поля ввода:
"ФИО плательщика"-->
    <text-field id="address" title="Адрес плательщика" max-len="254"
                keyboard="any:[ru,symb]:upper:true">
     <!--Регулярное выражение для валидации вводимых данных-->
     <validator type="regex">
       <rules> <rule regex="^.{3,254}$"/> </rules>
     </validator>
     <!--Подсказка, отображается на экране-->
     <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
    </text-field>
    <!--Поле выбора типа оплаты-->
    <selector-field id="type" title="Тип оплаты">
     <items type="static">
       <item value="21" title="Паспорт РФ"/>
       <item value="22" title="Заграничный паспорт"/>
       <item value="31" title="Паспорт иностранного гражданина"/>
     </items>
    </selector-field>
   </fields>
   <actions>
```

```
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <goto-action type="Next" title="Далее" target="2screen"/>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <goto-action type="Prev" title="Назад" target="previous"/>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
 <!--Создание экрана ввода данных с несколькими полями и навигацией-->
 <screen type="group/navi" decor="list" title ="Введите данные" id="2screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 20. Название поля
ввода: "Серия документа"-->
    <text-field id="seria" title="Серия документа" max-len="20"
                    keyboard="Digital">
     <!--Регулярное выражение для валидации вводимых данных-->
     <validator type="regex">
      <rules> <rule regex="^\d{1,20}$"/> </rules>
     </validator>
    <!--Подсказка, отображается на экране-->
     <help> <![CDATA[<html>Введите серию документа]]> </help>
    </text-field>
    <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 13. Название поля
ввода: "Номер документа"-->
    <text-field id="nomer" title="Номер документа" max-len="13"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules> <rule regex="^\d{1,13}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите номер документа]]> </help>
    </text-field>
    <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский, без возможности
смены, максимальное количество вводимых символов 254. Название поля ввода:
"Кем выдан"-->
    <text-field id="name" title="Кем выдан" max-len="254"
                keyboard="any:[ru,symb]:upper:true">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
```

```
      <rules> <rule regex="^.{1,254}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите наименование органа, выдавшего
                  документ]]></help>
    </text-field>
    <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 8. Название поля ввода:
"Дата выдачи документа"-->
    <text-field id="date" title="Дата выдачи документа" max-len="8"
                    keyboard="Digital">
      <!--Регулярное выражение для валидации вводимых данных-->
      <validator type="regex">
        <rules>
          <rule regex="^[0123]\d{1}(0[1-9]|1[0-2])(19|20)\d{2}$"/>
        </rules>
      </validator>
        <!--Подсказка, отображается на экране-->
        <help> <![CDATA[<html>Введите дату выдачи документа]]> </help>
    </text-field>
   </fields>
   <!--Создание навигации на экране-->
   <navigation>
    <!--Создание первой кнопки навигации-->
    <action type="navi1" title="Номер брони">
      <goto target="0screen"/>
    </action>
    <!--Создание второй кнопки навигации-->
    <action type="navi2" title="Данные плательщика">
      <goto target="1screen"/>
    </action>
    <!--Создание третьей кнопки навигации-->
    <action type="navi3" title="Паспортные данные" current="true">
      <goto target="2screen"/>
    </action>
   </navigation>
   <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="Далее">
      <!--Модификация параметра date-->
      <modify src-key="date" t-value="^(\d{2})(\d{2})(\d{4})$"
              t-value-title="^(\d{2})(\d{2})(\d{4})$"
              v-value="$1.$2.$3" v-value-title="$1.$2.$3" />
```

```
     <goto target="confirm"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <goto-action type="Prev" title="Назад" target="1screen"/>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
 <!--Создание экрана подтверждения данных платежа-->
 <screen type="confirm" decor="long" title="Подтверждение данных платежа"
         id="confirm">
   <!--Список атрибутов, отображаемых на экране подтверждения-->
  <fields list="id1,fio,address,type,seria,nomer,name,date"/>
  <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <goto-action type="Next" title="Далее" target="pay"/>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <goto-action type="Prev" title="Назад" target="2screen"/>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
</scenario>
```

Экран, соответствующий экрану с **id=«2screen»** с секцией навигации, приведен на рисунке 4.4.1.

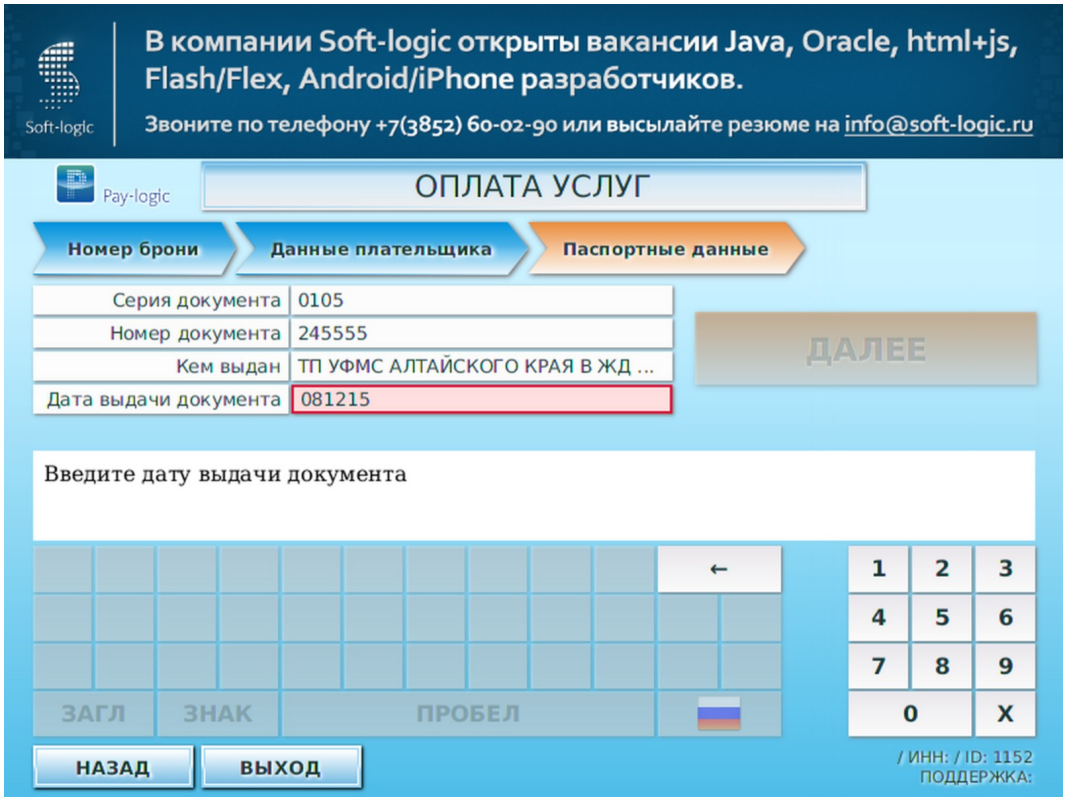


Рисунок 4.4.1 — Пример использования `<navigation>` с `<action>`

4.5 СОБЫТИЯ (<EVENTS>, <BARCODE-SCANNER>)

4.5.1 СОБЫТИЯ <EVENTS>

Для обработки событий, полученных от различных источников, используется элемент, описываемый тегом `<events>`. В текущей версии обрабатываются события от кард-ридера и сканера штрих-кодов.

Структура элемента:

```
<events>
      <event type=[CARD_STATE_INSIDE] action=[next|prev|exit]/>
</events>
```

Дочерний элемент: `<event>`.

Атрибуты: отсутствуют.

Атрибуты элемента `<event>`**: `type`, `action`.**

Таблица 4.5.1.1 — Атрибуты элемента `<event>`

Атрибут	Описание	Обяз.
`type`	Тип события, по наступлению которого необходимо выполнить действие. На данный момент есть только одно событие — **`CARD_STATE_INSIDE`** (карта успешно вставлена). Используется при работе со смарт-картами	Да
`action`	Действие, которое будет выполняться при	Да

Атрибут	Описание	Обяз.
	наступлении события. Возможные действия: **`Next`**, **`Prev`**, **`Exit`**. Соответствуют кнопкам экрана. Сами действия должны быть описаны внутри экрана	

Пример:

```
<!--Создание информационного экрана-->
<screen type="info" title="Вставьте карту" id="insert_screen">
   <field key="#info"/>
   <!--Тип события, при наступлении которого необходимо выполнить
действие-->
   <events>
     <event type="CARD_STATE_INSIDE" action="next"/>
   </events>
   <actions>
     <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
     <action type="Next" title="Далее">
       <!--Запрос к смарт-карте-->
       <hdw-request type="smart-card" function="validate" params="$all">
         <actions>
           <action type="success">
                 <set key="card_insert" value="1"/>
                 ...
           </action>
           ...
         </actions>
       </hdw-request>
     </action>
     <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
     <goto-action type="Prev" title="Назад" target="previous"/>
     <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
     <goto-action type="Exit" title="Выход" target="exit"/>
   </actions>
</screen>
```

В данном примере описан экран, на котором необходимо вставить смарт-карту. В тот момент, когда она будет успешно вставлена, сгенерируется событие и действие, описанное для кнопки «Далее», начнет автоматически выполняться.

4.5.2 НАЗНАЧЕНИЕ, СТРУКТУРА (<BARCODE-SCANNER>)

Элемент `<barcode-scanner>` обрабатывает сигнал со сканера штрих-кода для использования считанных данных на экране. Для работы со сканером не требуется специального экрана и поля. Структура элемента:

```
<barcode-scanner show-error=[true|false]
                 on-success={refresh|next]>
      <barcode-parser type=[regex|block]>
          <rules validate=<string>>
                <rule key=<string>
                      title=<string>
                      regex=<notEmptyString>
                      is-bind=[true|false]/>
          </rules>
      </barcode-parser>
</barcode-scanner>
```

Дочерние теги: `<barcode-parser>`.

Описание атрибутов элемента **<barcode-scanner>** приведено в таблице 4.5.2.1.

Таблица 4.5.2.1 — Описание атрибутов элемента **<barcode-scanner>**

Атрибут	Описание	Обяз.
show-error	Показывает диалог ошибки считывания штрих-кода, возможные значения: • **true** — диалог отображается; • **false** — диалог не отображается, значение по умолчанию	Нет
on-success	Тип события при успешном считывании данных со сканера штрих-кода	Нет

Доступны следующие события при успешном считывании данных со сканера штрих-кодов:

1. **`next`** — событие перехода на следующий экран.

2. **`refresh`** — событие обновления текущего экрана (есть возможность скорректировать отсканированные данные).

4.5.3 ПРАВИЛА РАСПОЗНАВАНИЯ ШТРИХ-КОДА (<BARCODE-PARSER>)

Элемент <barcode-parser> — определяет правила преобразования считанного штрих-кода для использования в качестве набора атрибутов платежа. Существуют два типа парсера, определяемые значением атрибута **type**:

1. **regex** — правила создания атрибутов определяются регулярным выражением.
2. **block** — позволяет формировать атрибуты платежа из части считанного значения.

Правила преобразования для элемента **<barcode-parser>** с типом **regex** определяются в дочерних элементах <rules>. В элементе <rules> может быть задан атрибут **validate**, который содержит регулярное выражение, определяющее допустимое значение штрих-кода. Например, если нужно работать только со штрих-кодами товаров, произведенных в России, требуется указать следующее значение **validate="^46\d.*"**, то есть все штрих-коды начинающиеся не с чисел 460-469 будут проигнорированы. Правила преобразования определяются в элементах <rule>.

Структура элемента:

```
<barcode-parser type="regex" >
      <rules validate=<string>>
          <rule  key=<string>
                 title=<string>
                 regex=<notEmptyString>
                 is-bind=[true|false]
                 convert-from=<integer>
                 convert-to=<integer>
                 flags=<flags_specification>/>
          ...
      </rules>
</barcode-parser>
```

Для элемента <barcode-parser> добавлен атрибут **disable-next=[true|false]**, который обрабатывается только при использовании на экране «INFO». Если указан

соответствующий флаг в сценарии/форме, то кнопка **«Далее»** на «INFO»-экране будет активна только после считывания штрих-кода при условии что **on-success=«refresh»**. На остальных типах экранов видимость кнопки регулируется валидацией полей ввода.

Описание атрибутов элемента `<rule>` приведено в таблице 4.5.3.1.

Таблица 4.5.3.1 — Описание атрибутов элемента `<rule>`

Атрибут	Описание	Обяз.
key	Ключ атрибута платежа, под которым будет сохранено значение полученного элемента	Да
title	Заголовок создаваемого атрибута платежа	Да
regex	Значение первой группы будет помещено в создаваемый атрибут платежа в качестве значения	Да
flags	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет
is-bind	Определяет обязательность создания данного атрибута платежа. Возможные значение: • **true** — создание элемента обязательно. Если элемент не был создан, выполнение сценария будет остановлено; • **false** — создание элемента необязательно. Выполнение сценария не будет остановлено, даже если элемент не был создан. В 5 версии ТПО **is_bind**	Нет
convert-from	Целочисленное значение, определяющее основание системы счисления полученного значения	Нет
convert-to	Целочисленное значение, определяющее основание системы счисления для результирующего значения	Нет

Пример:

```
<barcode-scanner>
  <barcode-parser type="regex" >
    <rules validate="^.*$">
      <rule key="id1" title="Л/с" regex="^(.*)$" is-bind="true"/>
      <rule key="id2" title="Период" regex="^(.*)$" is-bind="true"/>
    </rules>
  </barcode-parser>
</barcode-scanner>
```

Элемент **<barcode-parser>** с типом **block** позволяет передавать в атрибуты платежа часть считанного значения. Структура:

```
<barcode-parser type="block" >
      <rules validate=<string>>
          <rule  key=<string>
                 title=<string>
                 begin=<integer>
                 end=<integer>
                 flags=<flags_specification>/>
      </rules>
</barcode-parser>
```

Описание атрибутов элемента <rule> приведено в таблице 4.5.3.2.

Таблица 4.5.3.2 — Описание атрибутов элемента <rule>

Атрибут	Описание	Обяз.
key	Ключ атрибута платежа, под которым будет сохранено значение полученного элемента	Да
title	Заголовок создаваемого атрибута платежа	Да
begin	Целое число, определяет позицию первого символа выделяемой из штрих-кода подстроки	Да
end	Целое число, определяет позицию последнего символа выделяемой из штрих-кода подстроки, то есть символ, находящийся на позиции, указанной в параметре **end**, в подстроку не включается. Например, если указать	Да

Атрибут	Описание	Обяз.
	end=4, то в подстроку будет включен символ, находящий на 3 позиции, на 4 — не будет включен	
flags	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет

Пример:

```
<scenario begin="barcode" >
 <!--Создание информационного экрана-->
 <screen type="Info" title="QR-оплата" id="barcode">
  <field/>
  <!--Обработка сигнаал со сканера штрих-кодов для использования считанных
данных на экране-->
  <barcode-scanner show-error="true" on-success="next">
   <barcode-parser type="regex" >
    <rules validate="^.*$">
     <rule key="type" title="Тип" regex="^.*TYPE\=([^|]*)\|.*$"
           is-bind="true"/>
     <rule key="srv" title="Код услуги" regex="^.*SRV\=(\d*)\|.*$"
           is-bind="true"/>
     <rule key="id1" title="Номер счета" regex="^.*ID1\=([^|]*).*$"/>
    </rules>
   </barcode-parser>
  </barcode-scanner>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="ДАЛЕЕ">
    <!--Если srv ~ ^477$", то переброска на сервис с id 477-->
    <if condition="srv ~ ^477$">
     <then>
      <redirect target="477" params="$all"/>
     </then>
     <else>
      <!--Иначе: отображение информационного диалога с текстом ошибки-->
      <dialog type="Info" title="Ошибка"
              message="Невозможно определить вид услуги {0}" timeout="10"
```

```
            default="okay" params="id1">
      <actions>
       <!--Описание поведения сценария, если нажата кнопка "ОК"-->
       <action type="okay" title="ОК">
        <cancel/>
       </action>
      </actions>
     </dialog>
    </else>
   </if>
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
  <action type="Prev" title="НАЗАД">
   <goto target="previous"/>
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
  <action type="Exit" title="ВЫХОД">
   <goto target="exit"/>
  </action>
 </actions>
</screen>
</scenario>
```

4.6 ТИПЫ ЭКРАНОВ

4.6.1 ЭКРАН ОПЛАТЫ НЕСКОЛЬКИХ УСЛУГ (COMMUNAL)

Экран с типом **`type=«communal»`** предназначен для выбора нескольких услуг и общей оплаты по ним (рисунок 4.6.1.1).

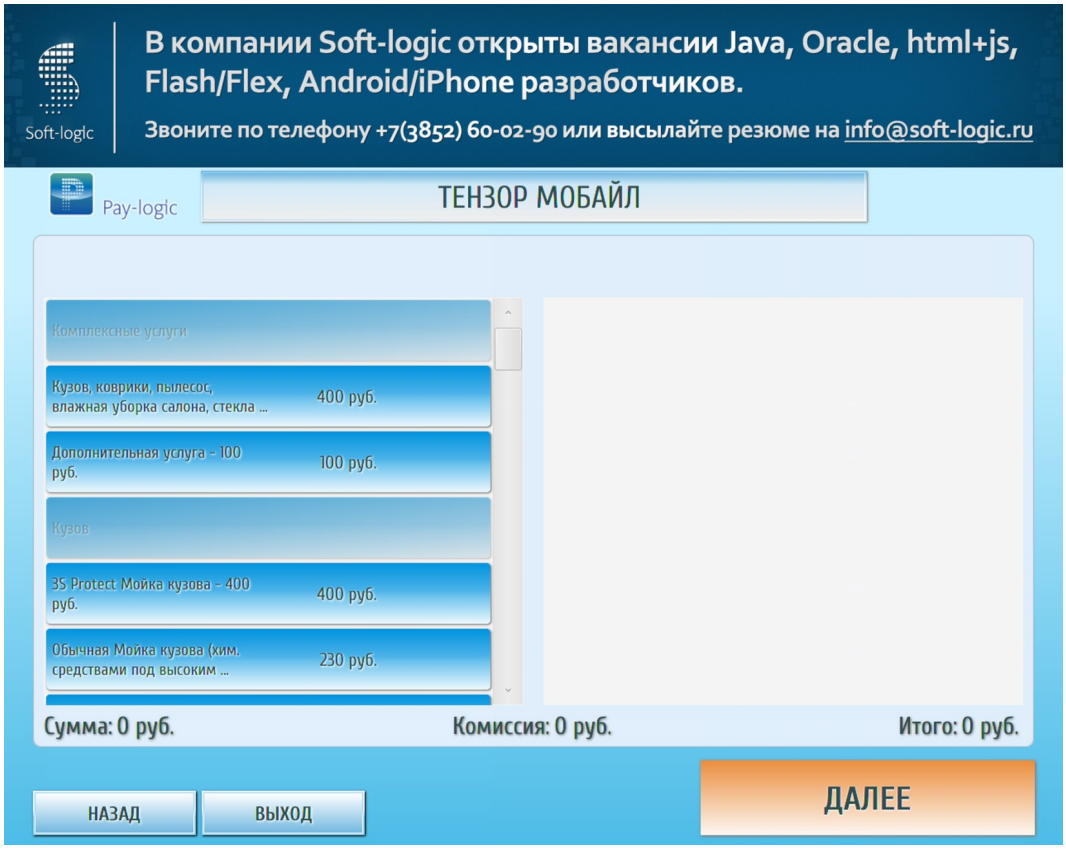


Рисунок 4.6.1.1 — Пример экрана COMMUNAL для интерфейса BlueSphere2

Для экрана оплаты нескольких услуг дополнительно доступен элемент `<fields>` с атрибутом **`key`**, который содержит название элемента в котором хранится nestedData, в

которой содержатся данные об элементах группового платежа. Поля ввода на данном типе экрана не используются. Работа с сложными структурами данных (nestedData), которыми оперирует экран, рассмотрена в разделе 7. Пример сценария, реализующего экран, приведен в разделе 7.1.

4.6.2 ЭКРАН КОРЗИНЫ ПЛАТЕЖЕЙ (COMMUNAL/BASKET)

Экран с типом **`type=«communal/basket»`** используется для групповых платежей, когда оплата по разным сервисам осуществляется в режиме корзины и по завершении выдается один чек. Поддерживается только на ТПО 5 в интерфейсе Smoke.

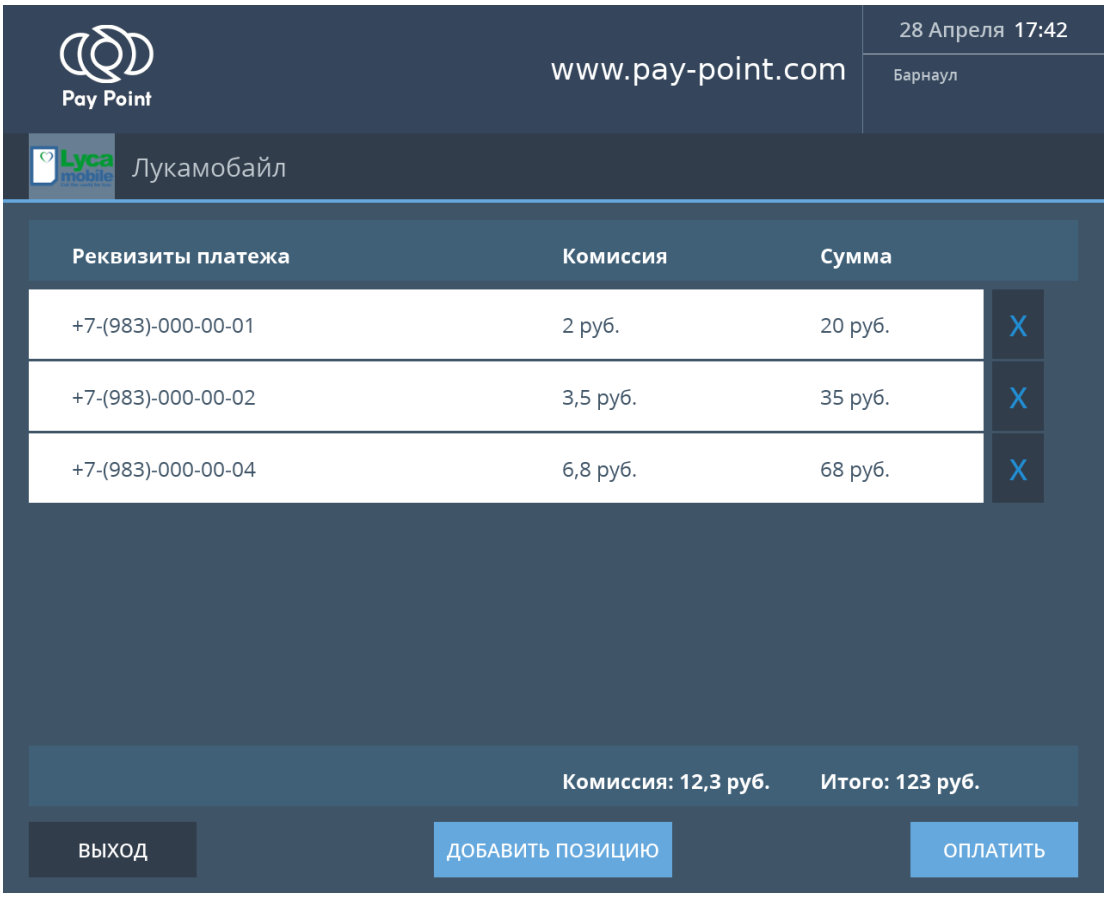


Рисунок 4.6.2.1 — Пример экрана COMMUNAL/BASKET для интерфейса Smoke

Чтобы задействовать данный экран, необходимо в сценарии в элементе `<payment-params>` указать атрибут **`use-basket="true"`** — в этом случае ТПО после выбора каждой услуги для оплаты будет сначала показывать экран корзины платежей с возможностью либо добавить платеж по другому сервису, либо перейти к единой оплате. Таким образом, оплата по сервису будет осуществляться не сразу.

Пример:

```
<payment-params key="#purchases" single-check="true" use-basket="true"
first-params-context="true"/>
```

4.6.3 ЭКРАН ПОЛУЧЕНИЯ СПИСКА УСЛУГ (COMMUNAL/CHARGE)

Экран с типом **`type=«communal/charge»`** используется для получения списка услуг для оплаты (рисунок 4.6.3.1). Сценарий, в котором отображается такой экран, использует корзину платежей.

Для получения показаний счетчиков используется онлайн-запрос:

```
<online-request type="default" function="get-meters-ls"
                params="id1,town-id,account-id" attempts="5"
                timeout="10" timeout-all="60">
```

В онлайн-запросе передаются параметры **`id1`**, **`town-id`**, **`account-id`** (значения данных параметров вводятся клиентом на предыдущих экранах, в данном примере не рассматриваются). Для получения данных используется функция `get-meters-ls`. Такой онлайн-запрос осуществляется в случае, если **`account-id`** удовлетворяет регулярному выражению **`^\d{1,25}$`**. В противном случае выполняется запрос:

```
<online-request type="default" function="get-meters-universal"
   params="id1,town-id,street-id,building-number,appartment-number"
   attempts="5" timeout="10" timeout-all="60">
```

В результате онлайн-запроса возвращается наименование сервиса и сумма к оплате. На данном экране можно выбрать оплачиваемые услуги, при этом атрибут **`selected`** data-элемента становится равным 1. Кроме того, можно изменить сумму к оплате, нажав кнопку «Редактировать».

Рисунок 4.6.3.1 — Экран получения списка услуг для оплаты (communal/charge)

Пример:

```
<!--Создание экрана получения списка услуг-->
<screen type="communal/charge" title="Выбор услуг" id="services_screen">
 <!--Отображение полей из объекта NestedData #services-->
 <fields key="#services"/>
 <actions>
 <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
 <action type="Next" title="ДАЛЕЕ">
  <!--Создание составного атрибута-->
  <text-format-table key="message1" template="${idservice}#${summ}#"
                     input-data="#services" default-value="0"
                     filter="selected=1"/>
  <!--Получение итоговой суммы группового платежа-->
  <sum key="summ" result-key="ssum1" input-data="#services"
       filter="selected=1"/>
```

```
 <!--Объявление переменной-->
 <set key="id1" value="1"/>
  <!--Проверка введенной суммы с учетом максимальной суммы по сервису и
комиссии-->
  <complex-validator type="purchase-validator" sum="ssum1">
   <actions>
    <!--Описание поведения сценария оплаты при успешном результате проверки"-->
    <action type="success">
     <!--Если #flag не равен null-->
     <if condition="is-not-null #flag">
      <then>
       <!--Удаление переменных из контекста-->
      <clear keys="#flag" />
      <goto target="meters_screen" />
     </then>
      <else>
      <!--Иначе: если значение account-id эквивалентено ^\d{1,25}$ -->
       <if condition="account-id ~ ^\d{1,25}$">
        <then>
         <!--Онлайн-запрос-->
         <online-request type="default" function="get-meters-ls"
                         params="id1,town-id,account-id" attempts="5"
                         timeout="10" timeout-all="60">
          <actions>
           <!--Если онлайн-запрос выполнен успешно-->
           <action type="success">
            <!--Удаление переменных из контекста-->
            <clear keys="id1" />
             <!--Переход на экран с id meters_screen-->
             <goto target="meters_screen" />
           </action>
           <!--Если онлайн-запрос завершился ошибкой-->
           <action type="error">
            <!--Отображение диалога с сообщением об ошибке-->
            <dialog type="Error" title="Информация"
                    message="Невозможно получить список счетчиков"
                    timeout="1" default="okay">
             <actions>
              <!--Описание поведения сценария, если нажата кнопка "OK"-->
             <action type="okay" title="OK">
              <!--Переход на экран с id request_error-->
              <goto target="request_error" />
             </action>
```

```
      </actions>
     </dialog>
     </action>
    <!--Если онлайн-запрос завершился исключением-->
     <action type="exception">
     <!--Отображение диалога-→
      <!--Отображение диалога с сообщением об ошибке-->
      <dialog type="Error" title="Error"
              message="Невозможно получить список счетчиков"
              timeout="1" default="okay">
       <actions>
        <!--Описание поведения сценария, если нажата кнопка "OK"-->
        <action type="okay" title="OK"> <cancel/> </action>
       </actions>
      </dialog>
     </action>
    </actions>
   </online-request>
  </then>
  <else>
   <!--Иначе: онлайн-запрос-->
   <online-request type="default" function="get-meters-universal"
       params="id1,town-id,street-id,building-number,appartment-number"
                  attempts="5" timeout="10" timeout-all="60">
    <actions>
     <!--Если онлайн-запрос выполнен успешно-->
     <action type="success">
      <!--Удаление переменной из контекста-->
      <clear keys="id1" />
      <goto target="meters_screen" />
     </action>
     <!--Если онлайн-запрос завершился ошибкой-->
     <action type="error">
      <!--Отображение диалога с сообщением об ошибке-->
      <dialog type="Error" title="Информация"
              message="Невозможно получить список счетчиков."
              timeout="1" default="okay">
       <actions>
        <!--Описание поведения сценария, если нажата кнопка "OK"-->
        <action type="okay" title="OK">
         <goto target="request_error" />
        </action>
       </actions>
```

```
      </dialog>
     </action>
      <!--Если онлайн-запрос завершился исключением-->
      <action type="exception">
      <!--Отображение диалога с сообщением об ошибке-->
      <dialog type="Error" title="Error"
              message="Невозможно получить список счетчиков"
              timeout="1" default="okay">
       <actions>
        <!--Описание поведения сценария, если нажата кнопка "OK"-->
        <action type="okay" title="OK"> <cancel/> </action>
       </actions>
      </dialog>
     </action>
    </actions>
   </online-request>
  </else>
 </if>
</else>
</if>
</action>
<!--Если онлайн-запрос завершился ошибкой-->
<action type="error">
 <!--Отображение диалога с сообщением об ошибке-->
 <dialog type="Error" title="Error"
         message="Общая сумма платежа превысила {0}. Скорректируйте."
         mess-params="max_purchase" timeout="10" default="okay">
  <actions>
   <!--Описание поведения сценария, если нажата кнопка "OK"-->
   <action type="okay" title="OK">
    <goto target="services_screen"/>
   </action>
  </actions>
 </dialog>
</action>
</actions>
</complex-validator>
<!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
<action type="Prev" title="НАЗАД">
 <!--Удаление переменной из контекста-->
 <clear keys="town-id" />
 <goto target="choose_town" />
</action>
```

```
<!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
<action type="Exit" title="ВЫХОД">
 <goto target="exit" />
</action>
<!--Описание поведения сценария оплаты при нажатии кнопки "Выбрать все"-->
<action type="SelectAll" title="Выбрать все">
 <!--Осуществляется циклический проход по набору элементов #services-->
 <for key="#services">
  <!--Распаковка набора элементов-->
  <unpack type="data" key="#for-element" elements="selected"/>
  <!--Объявление переменной-->
  <set key="selected" value="1" value-title="1" />
  <!--Упаковка набора элементов-->
  <pack type="data" key="#for-element" elements="selected"/>
 <!--Удаление переменной из контекста-->
  <clear keys="selected" />
 </for>
 <goto target="services_screen" />
</action>
<!--Описание поведения сценария оплаты при нажатии кнопки "Удалить все"-->
<action type="RemoveAll" title="Удалить все">
 <!--Осуществляется циклический проход по набору элементов #services-->
 <for key="#services">
  <unpack type="data" key="#for-element" elements="selected,summ,saldo"/>
  <!--Модификация параметра saldo-->
  <modify src-key="saldo" t-key="^(.*)$" t-value="^.{1}(.*$)"
          t-value-title="^.{1}(.*$)" v-key="summ" v-value="$1"
          v-value-title="$1"/>
  <!--Объявление переменной selected-->
  <set key="selected" value="0" value-title="0" />
  <!--Упаковка набора элементов-->
  <pack type="data" key="#for-element" elements="selected,summ,saldo"/>
  <!--Удаление переменной из контекста-->
  <clear keys="selected,summ,saldo"/>
 </for>
 <goto target="services_screen"/>
</action>
<!--Описание поведения сценария оплаты при нажатии кнопки "Редактировать"-->
<action type="Edit" title="Редактировать">
 <!--Если #selected не равно null-->
 <if condition="is-not-null #selected">
  <then>
   <!--Распаковка набора элементов-->
```

```
  <unpack type="data" key="#selected" elements="selected"/>
   <!--Если значение элемента selected равно 0-->
   <if condition="selected == 0">
    <then>
     <!--Распаковка набора элементов-->
     <unpack type="data" key="#selected" elements="name,saldo"/>
     <!--Модификация параметра saldo-->
     <modify src-key="saldo" t-key-title="^(.*)$" v-key-title="Сальдо"/>
     <goto target="input_sum"/>
    </then>
    <else>
     <!--Распаковка набора элементов-->
     <unpack type="data" key="#selected" elements="name,saldo"/>
     <!--Модификация параметра saldo-->
     <modify src-key="saldo" t-key-title="^(.*)$" v-key-title="Сальдо"/>
     <!--Переход на экран с id input_sum-->
     <goto target="input_sum"/>
    </else>
   </if>
  </then>
  <else>
   <!--Если #removed не равно null-->
   <if condition="is-not-null #removed">
    <then>
     <!--Распаковка набора элементов-→
     <unpack type="data" key="#removed" elements="summ,saldo"/>
     <!--Объявление переменной selected-->
     <set key="selected" value="0" value-title="0"/>
      <!--Модификация параметра saldo-->
     <modify src-key="saldo" t-key="^(.*)$" t-value="^.{1}(.*$)"
           t-value-title="^.{1}(.*$)" v-key="summ" v-value="$1"
           v-value-title="$1"/>
     <!--Упаковка набора элементов-->
     <pack type="data" key="#removed" elements="selected,summ,saldo"/>
     <!--Удаление переменных из контекста-->
     <clear keys="#removed,summ,saldo" />
     <!--Переход на экран с id services_screen-->
     <goto target="services_screen" />
    </then>
   <else/>
  </if>
 </else>
</if>
```

```
 </action>
 </actions>
</screen>
```

4.6.4 КОММУНАЛЬНЫЙ ЭКРАН (COMMUNAL/CHARGE/UNI)

Экран с типом **`type=«communal/charge/uni»`** (рисунок 4.6.4.1) предназначен для вывода списка услуг для ввода показаний. Доступен только в 7 версии ТПО.

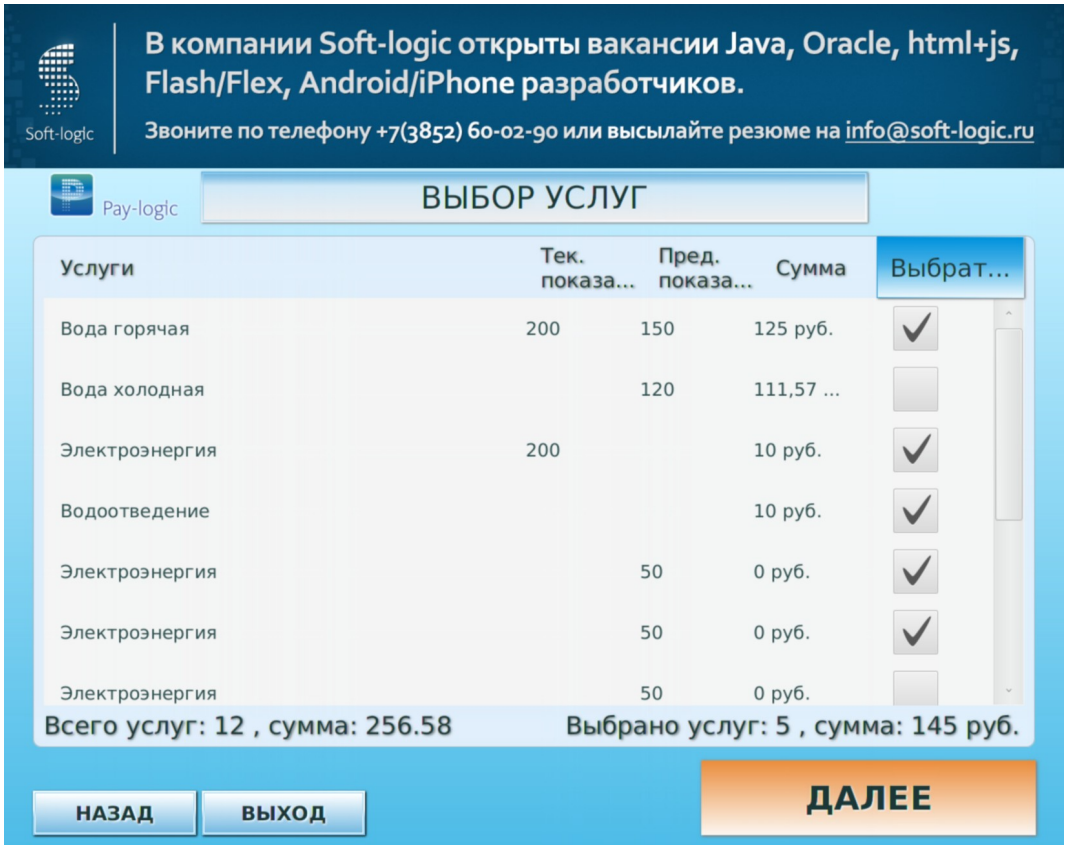


Рисунок 4.6.4.1 — Коммунальный экран (интерфейс Blues)

Выбор услуги к оплате/отказа от выбора осуществляется посредством установки/снятия чек-бокса.

В элементы корзины добавлен атрибут **`#infoIds`**, в котором через запятую указывается какие атрибуты будут отображены на экране.

Пример:

```
<set key="#infoIds" value="tariff,current" />
```

Максимальное количество отображаемых атрибутов определяется вёрсткой и на текущем экране в текущей версии отображаются два атрибута. Порядок атрибутов соответствует их порядку в **#infoIds**. В качестве заголовка колонки будет использоваться либо ключ атрибута, либо заголовок ключа (**key-title**). Если указаны оба, то будет использоваться **title**. Должен быть указан в каждой услуге, если указан для одной или нескольких, то будет использоваться значение **key**.

Пример:

```
<set key="tariff" value="2.5" key-title="Тариф"/>
```

Пример сценария с экраном communal/charge/uni и confirm/communal:

```
<scenario begin="init_screen">
 <!--Описание особенностей проведения группового платежа-->
 <payment-params key="#services" distribution-type="default"
                 single-check="true" />
 <!-- Экран формирования оффлайн данных  -->
 <screen type="Void" decor="simple" title ="" id="init_screen">
  <fields/>
  <actions>
   <action type="Next" title="Далее">
    <!--Объявление переменных-->
    <set key="#id" value="s1"/>
    <set key="name" value="Вода горячая" key-title="Наименование услуги"/>
    <set key="current" value="200" key-title="Тек. показания"/>
    <set key="previous" value="150" key-title="Пред. Показания"/>
    <set key="tariff" value="2.11" key-title="Тариф"/>
    <set key="consumption" value="50" key-title="Потребление"/>
    <set key="summ" value="125" key-title="Сумма"/>
    <set key="selected" value="1"/>
    <set key="#infoIds" value="current,previous" />
    <set key="allow-change-state" value="0" key-title="0"/>
<!--Упаковка набора элементов-->
    <pack type="data" key="dt1" elements="#id,name,current,previous,tariff,
                                          consumption,selected,summ,#infoIds"/>
    <!--Объявление переменных-->
    <set key="#id" value="s2"/>
```

```
<set key="name" value="Вода холодная" key-title="Наименование услуги"/>
<set key="current" value="" key-title="Тек. показания"/>
<set key="previous" value="120" key-title="Пред. показания"/>
<set key="tariff" value="1.5" key-title="Тариф"/>
<set key="consumption" value="" key-title="Потребление"/>
<set key="summ" value="111.57851800" key-title="Сумма"/>
<set key="selected" value="0"/>
<set key="editable" value="0"/>
<set key="#infoIds" value="current,previous" />
<!--Упаковка набора элементов-->
<pack type="data" key="dt2" elements="#id,name,current,previous,tariff,
                        consumption,selected,summ,#infoIds,editable"/>
<!--Объявление переменных-->
<set key="#id" value="s3"/>
<set key="name" value="Электроэнергия" key-title="Наименование услуги"/>
<set key="current" value="200" key-title="Тек. Показания"/>
<set key="previous" value="" key-title="Пред. Показания"/>
<set key="tariff" value="5.25" key-title="Тариф"/>
<set key="consumption" value="" key-title="Потребление"/>
<set key="summ" value="10" key-title="Сумма"/>
<set key="selected" value="0"/>
<set key="#infoIds" value="current,previous" />
<!--Упаковка набора элементов-→
<pack type="data" key="dt3" elements="#id,name,current,previous,tariff,
                                    consumption,selected,summ,#infoIds"/>
<!--Объявление переменных-->
<set key="#id" value="s4"/>
<set key="name" value="Водоотведение" key-title="Наименование услуги"/>
<set key="current" value="" key-title="Тек. показания"/>
<set key="tariff" value="1.1" key-title="Тариф"/>
<set key="consumption" value="" key-title="Потребление"/>
 <set key="summ" value="10" key-title="Сумма"/>
<set key="selected" value="0"/>
<set key="#infoIds" value="current,previous" />
<!--Упаковка набора элементов-->
<pack type="data" key="dt4" elements="#id,name,current,previous,tariff,
                                  consumption,selected,summ,#infoIds"/>
<!--Объявление переменных-->
<set key="#id" value="s5"/>
<set key="name" value="Электроэнергия" key-title="Наименование услуги"/>
<set key="current" value="" key-title="Тек. показания"/>
<set key="previous" value="50" key-title="Пред. показания"/>
<set key="tariff" value="5.25" key-title="Тариф"/>
```

```
<set key="consumption" value="" key-title="Потребление"/>
<set key="summ" value="" key-title="Сумма"/>
<set key="selected" value="0"/>
<set key="#infoIds" value="current,previous" />
<!--Упаковка набора элементов-->
<pack type="data" key="dt5" elements="#id,name,current,previous,tariff,
                                      consumption,selected,summ,#infoIds"/>
<!--Объявление переменных-->
<set key="#id" value="s6"/>
<set key="name" value="Электроэнергия" key-title="Наименование услуги"/>
<set key="current" value="" key-title="Тек. показания"/>
<set key="previous" value="50" key-title="Пред. показания"/>
<set key="tariff" value="5.25" key-title="Тариф"/>
<set key="consumption" value="" key-title="Потребление"/>
<set key="summ" value="" key-title="Сумма"/>
<set key="selected" value="0"/>
<set key="#infoIds" value="current,previous" />
<!--Упаковка набора элементов-->
<pack type="data" key="dt6" elements="#id,name,current,previous,tariff,
                                      consumption,selected,summ,#infoIds"/>
<!--Объявление переменных-->
<set key="#id" value="s7"/>
<set key="name" value="Электроэнергия" key-title="Наименование услуги"/>
<set key="current" value="" key-title="Тек. Показания"/>
<set key="previous" value="50" key-title="Пред. показания"/>
<set key="tariff" value="5.25" key-title="Тариф"/>
<set key="consumption" value="" key-title="Потребление"/>
<set key="summ" value="" key-title="Сумма"/>
<set key="selected" value="0"/>
<set key="#infoIds" value="current,previous" />
<!--Упаковка набора элементов-->
<pack type="data" key="dt7" elements="#id,name,current,previous,tariff,
                                      consumption,selected,summ,#infoIds"/>
<!--Объявление переменных-->
<set key="#id" value="s8"/>
<set key="name" value="Электроэнергия" key-title="Наименование услуги"/>
<set key="current" value="" key-title="Тек. показания"/>
<set key="previous" value="50" key-title="Пред. показания"/>
<set key="tariff" value="5.25" key-title="Тариф"/>
<set key="consumption" value="" key-title="Потребление"/>
<set key="summ" value="" key-title="Сумма"/>
<set key="selected" value="0"/>
<set key="#infoIds" value="current,previous" />
```

```
<!--Упаковка набора элементов-->
<pack type="data" key="dt8" elements="#id,name,current,previous,tariff,
                                    consumption,selected,summ,#infoIds"/>
<!--Объявление переменных-->
<set key="#id" value="s9"/>
<set key="name" value="Электроэнергия" key-title="Наименование услуги"/>
<set key="current" value="" key-title="Тек. показания"/>
<set key="previous" value="50" key-title="Пред. показания"/>
<set key="tariff" value="5.25" key-title="Тариф"/>
<set key="consumption" value="" key-title="Потребление"/>
<set key="summ" value="" key-title="Сумма"/>
<set key="selected" value="0"/>
<set key="#infoIds" value="current,previous" />
<!--Упаковка набора элементов-->
<pack type="data" key="dt9" elements="#id,name,current,previous,tariff,
                                    consumption,selected,summ,#infoIds"/>
<!--Объявление переменных-->
<set key="#id" value="s10"/>
<set key="name" value="Электроэнергия" key-title="Наименование услуги"/>
<set key="current" value="" key-title="Тек. показания"/>
<set key="previous" value="50" key-title="Пред. Показания"/>
<set key="tariff" value="5.25" key-title="Тариф"/>
<set key="consumption" value="" key-title="Потребление"/>
<set key="summ" value="" key-title="Сумма"/>
<set key="selected" value="0"/>
<set key="#infoIds" value="current,previous" />
<!--Упаковка набора элементов-->
<pack type="data" key="dt10" elements="#id,name,current,previous,tariff,
                                    consumption,selected,summ,#infoIds"/>
<!--Объявление переменных-->
<set key="#id" value="s11"/>
<set key="name" value="Электроэнергия" key-title="Наименование услуги"/>
<set key="current" value="" key-title="Тек. показания"/>
<set key="previous" value="50" key-title="Пред. показания"/>
<set key="tariff" value="5.25" key-title="Тариф"/>
<set key="consumption" value="" key-title="Потребление"/>
<set key="summ" value="" key-title="Сумма"/>
<set key="selected" value="0"/>
<set key="#infoIds" value="current,previous" />
<!--Упаковка набора элементов-->
<pack type="data" key="dt11" elements="#id,name,current,previous,tariff,
                                    consumption,selected,summ,#infoIds"/>
<!--Объявление переменных-->
```

```
    <set key="#id" value="s12"/>
    <set key="name" value="Электроэнергия" key-title="Наименование услуги"/>
    <set key="current" value="" key-title="Тек. показания"/>
    <set key="previous" value="50" key-title="Пред. показания"/>
    <set key="tariff" value="5.25" key-title="Тариф"/>
    <set key="consumption" value="" key-title="Потребление"/>
    <set key="summ" value="" key-title="Сумма"/>
    <set key="selected" value="0"/>
    <set key="#infoIds" value="current,previous" />
    <!--Упаковка набора элементов-→
    <pack type="data" key="dt12" elements="#id,name,current,previous,tariff,
                                          consumption,selected,summ,#infoIds"/>
    <pack type="nested" key="#services" elements="dt1,dt2,dt3,dt4,dt5,dt6,
                                                  dt7,dt8,dt9,dt10,dt11,dt12"/>
    <!--Удаление переменных из контекста-->
    <clear keys="#id,name,current,previous,tariff,consumption,
                 selected,summ,dt1,dt2,dt3,dt4,dt5,dt6,dt7,dt8,dt9,dt10,
                 dt11,dt12"/>
    <!--Объявление переменной id1-->
    <set key="id1" value="123456789"/>
    <goto target="com_screen"/>
   </action>
  </actions>
 </screen>
 <!--Создание экрана ввода показаний по коммунальным услугам-->
 <screen type="Communal/charge/uni" decor="simple" title="Выбор услуг"
         id="com_screen">
  <!--Отображение полей из объекта NestedData #services-->
  <fields key="#services"/>
  <actions>
   <action type="Next" title="">
    <!--Получение итоговой суммы группового платежа-->
    <sum key="summ" result-key="ssum" input-data="#services"
         filter="selected=1"/>
    <!--Передача значения числовой переменной в сумму платежа без учета
комиссии-->
    <set-sum from="ssum" type="units"/>
    <!--Удаление переменных из контекста-→
    <clear keys="ssum"/>
    <goto target="input_fio"/>
   </action>
   <action type="Prev" title="">
    <goto target="first"/>
```

```
  </action>
  <action type="Exit" title="">
   <goto target="exit"/>
  </action>
  <action type="SelectAll" title="Выбрать все">
   <!--Осуществляется циклический проход по набору элементов #services-->
   <for key="#services">
    <!--Объявление переменной selected-->
    <set key="selected" value="1" value-title="1" />
    <!--Упаковка набора элементов-->
    <pack type="data" key="#for-element" elements="selected" />
    <!--Удаление переменных из контекста-->
    <clear keys="selected" />
   </for>
   <goto target="com_screen" />
  </action>
  <action type="RemoveAll" title="Удалить все">
   <!--Осуществляется циклический проход по набору элементов #services-->
   <for key="#services">
    <!--Объявление переменной selected-->
    <set key="selected" value="0" value-title="0" />
    <pack type="data" key="#for-element" elements="selected" />
    <!--Удаление переменных из контекста-→
    <clear keys="selected" />
   </for>
   <goto target="com_screen" />
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Редактировать"--
>
  <action type="Edit" title="Редактировать">
   <!--Если #selected не равно null-->
   <if condition="is-not-null #selected">
    <then>
     <!--Объявление переменной selected-->
     <set key="selected" value="1" value-title="1" />
     <!--Упаковка набора элементов-->
     <pack type="data" key="#selected" elements="selected" />
     <!--Удаление переменных из контекста-->
     <clear keys="#selected,selected" />
     <goto target="com_screen" />
    </then>
    <else>
     <!--Если #removed не равно null-->
```

```
     <if condition="is-not-null #removed">
      <then>
       <!--Объявление переменной selected-->
       <set key="selected" value="0" value-title="0" />
       <pack type="data" key="#removed" elements="selected" />
       <!--Удаление переменных из контекста-->
       <clear keys="#removed,selected" />
       <goto target="com_screen" />
      </then>
      <else>
       <!--Иначе: если #removed не равно null-→
       <if condition="is-not-null #edit">
        <then>
         <!--Распаковка набора элементов-->
         <unpack type="data" key="#edit" elements="summ,consumption,name,
                                              previous,current,tariff" />
         <goto target="input_sum" />
        </then>
        <else>
         <goto target="com_screen" />
        </else>
       </if>
      </else>
     </if>
    </else>
   </if>
  </action>
 </actions>
</screen>

<!--Создание группового экрана ввода данных-->
<screen type="group" title="Укажите сумму услуги" id="input_sum">
 <fields>
  <!--Создание поля ввода числа в десятичном формате через точку. Название
поля ввода: "Пред. показания"-->
  <numeric-field id="previous" title="Пред. показания" format="5.2">
   <verify>
    <range begin="1" end="15000" />
   </verify>
   <!--Подсказка, отображается на экране-->
   <help>
    <![CDATA[Введите текущие показания приборов учета за прошлый период]]>
   </help>
```

```
   </numeric-field>
   <!--Создание поля ввода числа в десятичном формате через точку. Название
поля ввода: "Тек. показания"-->
   <numeric-field id="current" title="Тек. показания" format="5.2">
    <verify>
     <range begin="1" end="15000" />
    </verify>
    <!--Подсказка, отображается на экране-->
    <help>
     <![CDATA[Введите текущие показания приборов учета]]>
    </help>
   </numeric-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="ДАЛЕЕ">
    <!--Выполнение математических действий-->
    <math operation="current - previous" result-key="consumption"
                                              result-title="Потреблено"/>
    <math operation="consumption * tariff" result-key="summ"
                                                  result-title="Сумма"/>
    <!--Упаковка набора элементов-->
    <pack type="data" key="#edit" elements="summ,consumption,current,
                                           previous" />
    <!--Отображение диалога с информационным сообщением-->
    <dialog type="Info" title="Проверка переменных"
           message="Пред. показания: {0} Тек. показания: {1} Разница: {2}
                    Тариф: {3} Сумма: {4}"
           mess-params="previous,current,consumption,tariff,summ"
           timeout="10" default="okay">
     <actions>
      <!--Описание поведения сценария, если нажата кнопка "OK"-->
      <action type="okay" title="OK">
       <!--Удаление переменных из контекста-->
       <clear keys="#edit,summ,consumption,name" />
       <goto target="input_fio" />
      </action>
     </actions>
    </dialog>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="НАЗАД">
    <!--Удаление переменных из контекста-->
```

```
    <clear keys="#edit,summ,consumption,name" />
    <goto target="com_screen" />
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="ВЫХОД">
    <goto target="exit" />
   </action>
  </actions>
 </screen>

 <!--Создание группового экрана выбора данных-->
 <screen type="group" title ="Введите данные" id="input_fio">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский, без возможности
смены, максимальное количество вводимых символов 60. Название поля ввода:
"ФИО плательщика"-->
   <text-field id="fio" title="ФИО плательщика" max-len="60"
               keyboard="any:[ru,symb]:upper:true">
    <!--Регулярное выражение для валидации введенных данных-->
    <validator type="regex">
     <rules>
      <rule regex="^[а-яА-Яёё-]{2,20}[\s]{1}[а-яА-Яёё-]{2,20}[\s]{1,3}
                  [а-яА-ЯЁё-]{2,20}([\s]{1}[а-яА-ЯЁё-]{2,20})?$"/>
     </rules>
    </validator>
     <!--Подсказка, отображается на экране-->
     <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский, без возможности
смены, максимальное количество вводимых символов 254. Название поля ввода:
"Адрес плательщика"-→
   <text-field id="address" title="Адрес плательщика" max-len="254"
               keyboard="any:[ru,symb]:upper:true">
    <!--Регулярное выражение для валидации введенных данных-->
    <validator type="regex">
     <rules> <rule regex="^.{3,254}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
   </text-field>
  </fields>
```

```
  <actions
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <goto-action type="Next" title="Далее" target="new_confirm"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <goto-action type="Prev" title="Назад" target="previous"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>

 <!--Создание экрана подтверждения выбранных услуг-->
 <screen type="confirm/communal" title="ВЫБРАННЫЕ УСЛУГИ" id="new_confirm">
 <!--Список атрибутов, отображаемых на экране подтверждения-→
  <fields list="id1,fio,address,#services"/>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <goto target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="com_screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-→
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>

</SCENARIO>
```

4.6.5 ЭКРАН ЖКХ (COMMUNAL/METER)

Экран с типом **`type=«communal/meter»`** предназначен для ввода показаний счетчиков и приборов учета (рисунок 4.6.5.1). Для экрана ЖКХ дополнительно доступен элемент `<fields>` с атрибутом **`key`**, который содержит название элемента в котором хранится nestedData, в которой содержатся данные о показаниях счетчиков. Поля ввода на данном типе экрана не используются. Работа с сложными структурами данных (nestedData), которыми оперирует экран, рассмотрена в разделе 7.

Рисунок 4.6.5.1 — Пример экрана COMMUNAL/METER для интерфейса Blues

4.6.6 ЭКРАН ПОДТВЕРЖДЕНИЯ (CONFIRM)

Экран с типом **`type=«confirm»`** представляет собой экран подтверждения введенной ранее информации (рисунки 4.6.6.1, 4.6.6.2). Для экрана подтверждения дополнительно доступен элемент `<fields>` с атрибутом **`list`**, который содержит список названий элементов, которые нужно отобразить на текущем экране подтверждения. Например: **`«list=key1, key2, key3»`**. Поля ввода на данном типе экрана не используются.

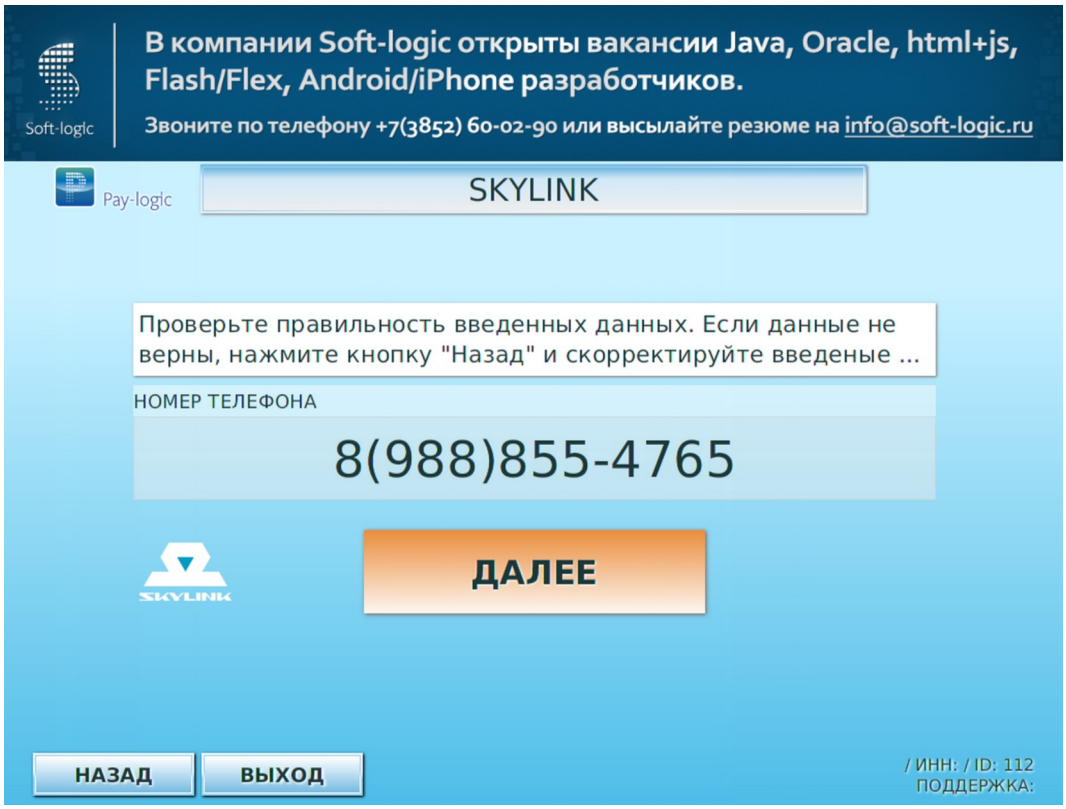


Рисунок 4.6.6.1 — Пример экрана CONFIRM для интерфейса Blues

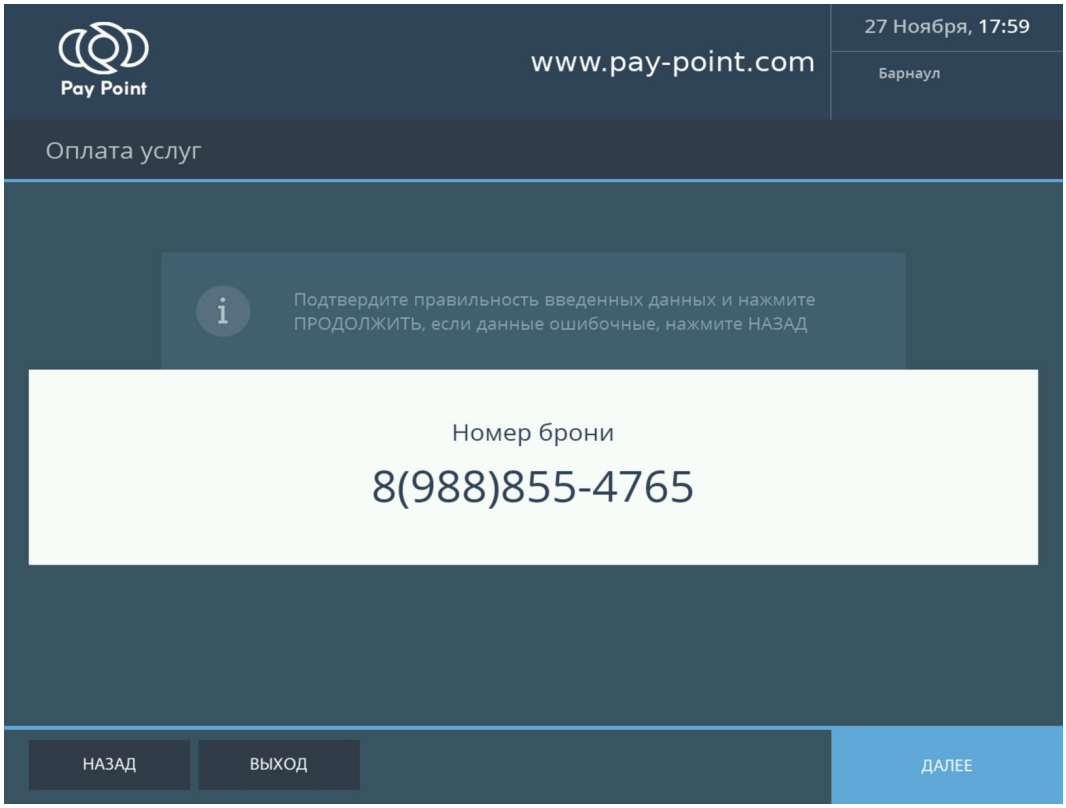


Рисунок 4.6.6.2 — Пример экрана CONFIRM для интерфейса Smoke

В 5 версии ТПО экран подтверждения может отображать до 8-ми полей. В 7 версии ТПО на стандартном экране подтверждения отображается 1 поле. Для отображения нескольких полей необходимо использовать **`type=«confirm/multiple»`**.

На экране подтверждения `<fields>` (раздел 5.1) используется с указанием атрибутов:

```
<screen type="confirm"
        ...>
           <fields list=<keyListType>/>
                    ...
</screen>
```

Атрибуты: `list`, `title`.

На экране «CONFIRM» необходимо указать атрибуты:

1. **`list`** — список идентификаторов атрибутов платежа, которые нужно отобразить на экране. Следует учитывать тип экрана подтверждения и тип пользовательского интерфейса, так как отображается только ограниченное кол-во атрибутов. Все не вошедшие атрибуты будут проигнорированы. Информация выводится на экран в режиме просмотра. Непустая строка до 30-ти символов.

В 5 версии ТПО в качестве заголовка экрана отображается название сервиса, в 7 версии — значение атрибута **`title`** экрана.

4.6.7 ЭКРАН ПОДТВЕРЖДЕНИЯ ДО 3 ПОЛЕЙ (CONFIRM-BIG)

Экран с типом **`type=«confirm-big»`** в ТПО 5 и 7 версий визуально аналогичен типу «CONFIRM», но может отображать до трех полей. Внешний вид экрана приведен на рисунке 4.6.7.1. На экране используется `<fields>` с указанием атрибута **`list`**, который содержит список идентификаторов атрибутов платежа, отображаемых на экране.

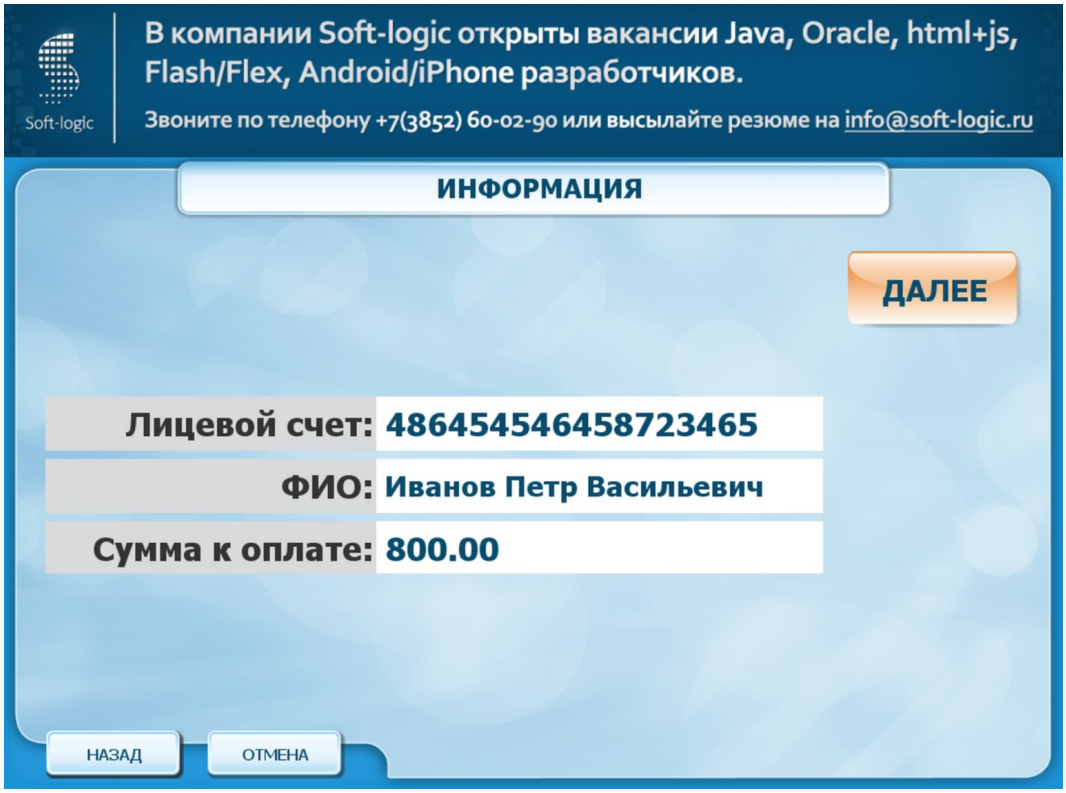


Рисунок 4.6.7.1 — Пример экрана CONFIRM-BIG для интерфейса BlueSphere2

4.6.8 ЭКРАН ПОДТВЕРЖДЕНИЯ С ОТОБРАЖЕНИЕМ ИНФОРМАЦИИ О КОМИССИИ (CONFIRM/COMMISSION)

Экран **`type=«confirm/commission»`** представляет собой экран подтверждения введенной ранее информации с отображением информации о комиссии по сервису (рисунок 4.6.8.1).

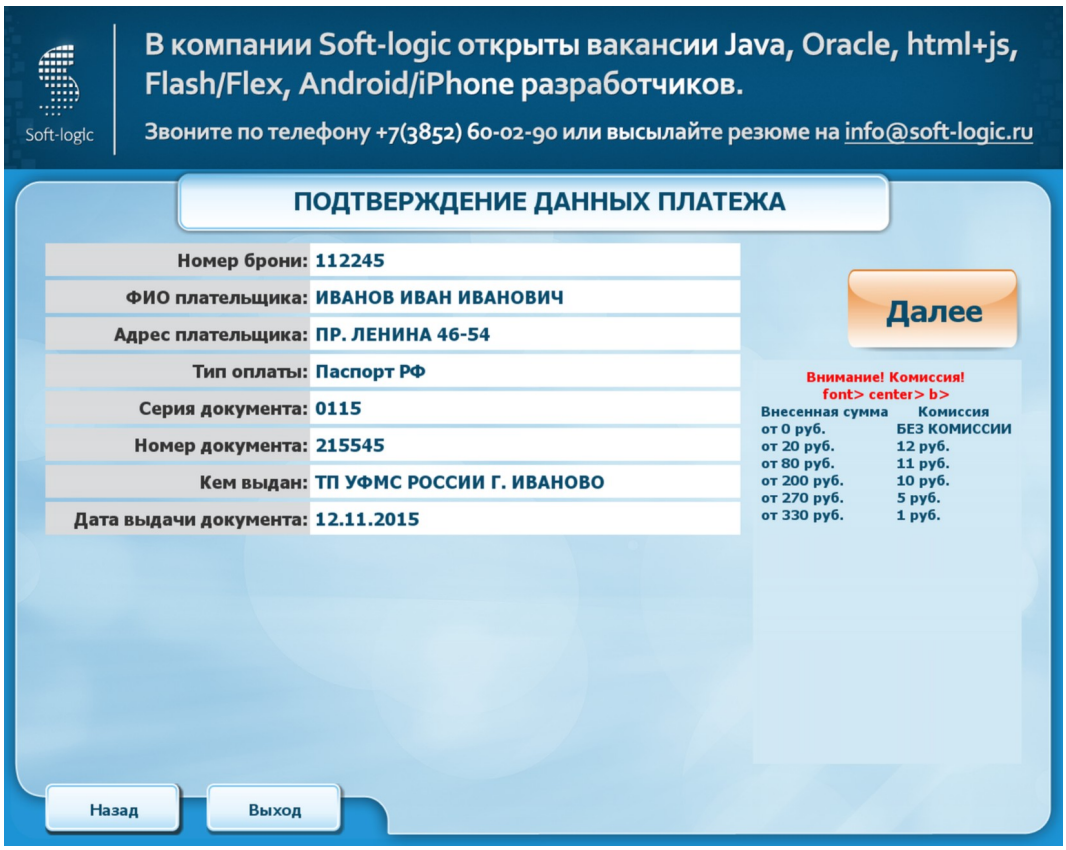


Рисунок 4.6.8.1 — Пример экрана CONFIRM/COMMISSION для интерфейса BlueSphere2

Пример (для 5 (5im) версии ТПО):

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!--Создание экрана ввода номера брони-->
 <screen type="numeric" title ="Введите номер брони" id="0screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 6. Название поля
ввода: "Номер брони"-->
   <text-field id="id1" title="Номер брони" max-len="6"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules> <rule regex="^.{6}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите номер брони]]> </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <goto target="1screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="previous"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
 <!--Создание группового экрана ввода данных -->
 <screen type="group" title ="Введите данные"
         id="1screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура с возможностью переключения языка,
максимальное количество вводимых символов 60. Название поля ввода: "ФИО
плательщика"-->
   <text-field id="fio" title="ФИО плательщика" max-len="60"
```

```
                keyboard="any:ru:en:symb:upper:true">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules>
      <rule regex="^[a-яA-Яёё-]{2,20}[\s]{1}[a-яA-Яёё-]{2,20}[\s]{1,3}
            [a-яA-ЯЁё-]{2,20}([\s]{1}[a-яA-ЯЁё-]{2,20})?$"/>
     </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура с возможностью переключения языка,
максимальное количество вводимых символов 254. Название поля ввода:
"Адрес плательщика"-->
   <text-field id="address" title="Адрес плательщика" max-len="254"
                keyboard="any:ru:en:symb:upper:true">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules> <rule regex="^.{3,254}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
   </text-field>
   <!--Создание поля выбора типа оплаты-->
   <selector-field id="type" title="Тип оплаты">
    <items type="static">
     <item value="21" title="Паспорт РФ"/>
     <item value="22" title="Заграничный паспорт"/>
     <item value="31" title="Паспорт иностранного гражданина"/>
    </items>
   </selector-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <goto target="2screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="previous"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
```

```
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
 <!--Создание группового экрана для ввода данных-->
 <screen type="group" title ="Введите данные"
         id="2screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 20. Название поля
ввода: "Серия документа"-->
   <text-field id="seria" title="Серия документа" max-len="20"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules> <rule regex="^\d{1,20}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите серию документа]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 13. Название поля
ввода: "Номер документа"-->
   <text-field id="nomer" title="Номер документа" max-len="13"
               keyboard="Digital">
   <!--Регулярное выражение для валидации вводимого номера-->
   <validator type="regex">
    <rules> <rule regex="^\d{1,13}$"/> </rules>
   </validator>
   <!--Подсказка, отображается на экране-->
   <help> <![CDATA[<html>Введите номер документа]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура с возможностью переключения языка,
максимальное количество вводимых символов 254. Название поля ввода: "Кем
выдан"-→
   <text-field id="name" title="Кем выдан" max-len="254"
               keyboard="any:ru:en:symb:upper:true">
   <!--Регулярное выражение для валидации вводимого номера-->
   <validator type="regex">
    <rules> <rule regex="^.{1,254}$"/> </rules>
   </validator>
```

```
   <!--Подсказка, отображается на экране-->
   <help>
    <![CDATA[<html>Введите наименование органа, выдавшего документ]]>
   </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 8. Название поля
ввода: "Дата выдачи документа"-->
   <text-field id="date" title="Дата выдачи документа" max-len="8"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules>
      <rule regex="^[0123]\d{1}(0[1-9]|1[0-2])(19|20)\d{2}$"/>
     </rules>
    </validator>
     <!--Подсказка, отображается на экране-->
     <help> <![CDATA[<html>Введите дату выдачи документа]]> </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
   <!--Модификация параметра date-->
    <modify src-key="date" t-value="^(\d{2})(\d{2})(\d{4})$"
           t-value-title="^(\d{2})(\d{2})(\d{4})$"
           v-value="$1.$2.$3" v-value-title="$1.$2.$3" />
    <goto target="confirm"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="1screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
 <!--Создание экрана подтверждения-->
 <screen type="confirm" decor="commission"
         title ="Подтверждение данных платежа" id="confirm">
  <!--Список атрибутов, отображаемых на экране подтверждения-->
```

```
  <fields list="id1,fio,address,type,seria,nomer,name,date"/>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <goto  target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="2screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
</scenario>
```

4.6.9 ЭКРАН ПОДТВЕРЖДЕНИЯ В ВИДЕ ТАБЛИЧНОГО СЧЕТА (CONFIRM/COMMUNAL)

Экран с типом **type=«confirm/communal»** предназначен для вывода данных на экран подтверждения в виде квитанции. Доступен только в 7 версии ТПО.

Внутри экрана возможно указать элемент `<fields list ...>` с перечнем отображаемых атрибутов. Из указанных в `<fields list ...>` атрибутов будет составлен список для отображения на экране. В первую очередь экран обработает обычные InputElement'ы, то есть в примере **id1**, **fio**, **address**, затем в список отображаемых на экране данных добавятся элементы корзины. Таким образом, порядок идентификаторов в атрибуте **list** не влияет на расположение услуг и InputElement-ов относительно друг друга. Элементы корзины всегда будут в конце.

Пример (рисунок 4.6.9.1): **id1, fio, address** — InputElement-ы, **#services** — корзина.

В сценарии указываем секцию `<fields list="id1,fio,address,#services"/>` или `<fields list="#services,id1,fio,address"/>`**.** Так же как учитывается **#infoIds**. На данном экране отображается 2 таких атрибута (ограничения верстки экрана). Пример сценария приведен в разделе 4.6.4.

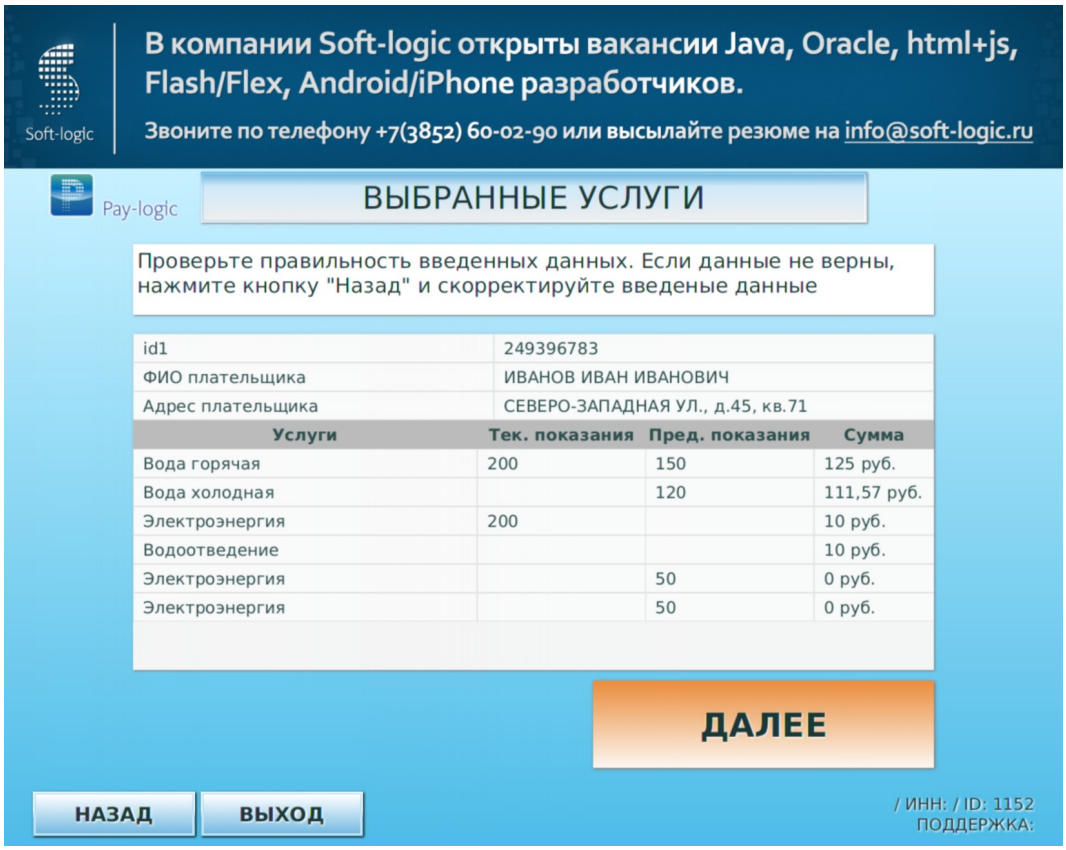


Рисунок 4.6.9.1 — Экран подтверждения в виде табличного счета (Blues)

4.6.10 ЭКРАН ПОДТВЕРЖДЕНИЯ (CONFIRM/LONG)

Экран **`type=«confirm/long»`** представляет собой незначительно модифицированный экран подтверждения введенной ранее информации (рисунок 4.6.10.1). Стандартный экран «CONFIRM» приведен на рисунке 4.6.23.2.

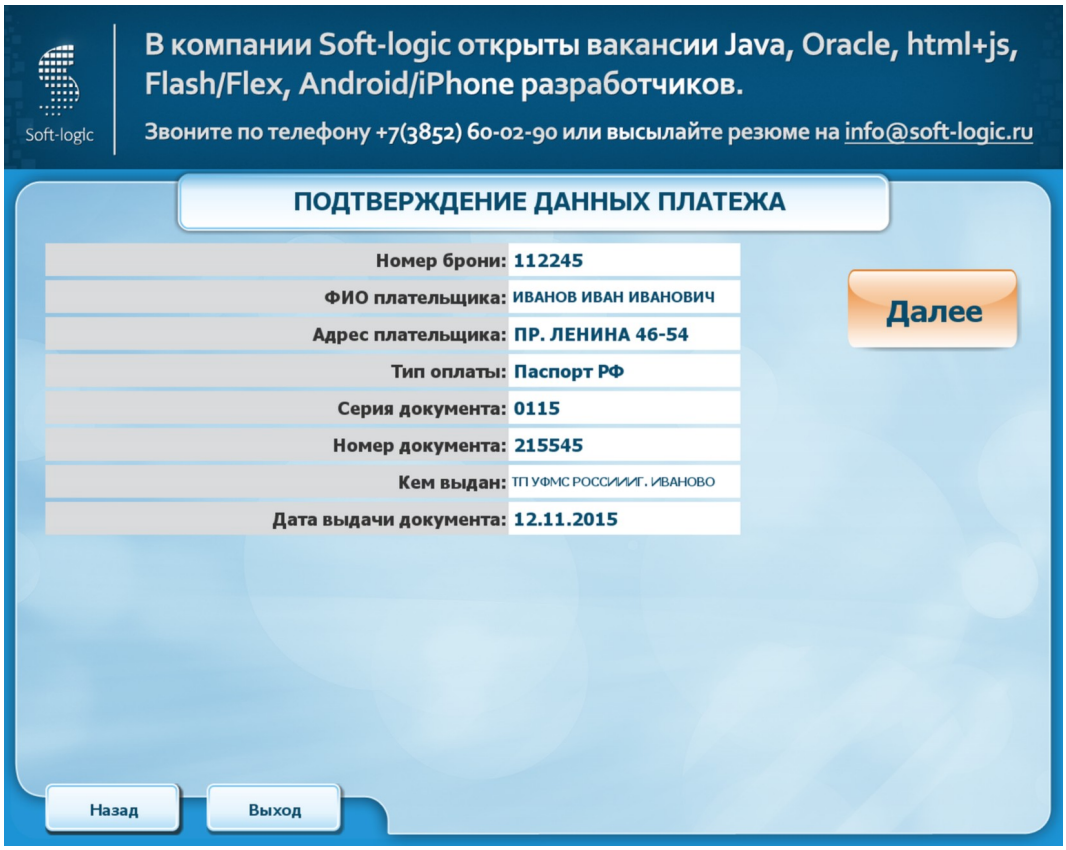


Рисунок 4.6.10.1 — Пример экрана CONFIRM/LONG для интерфейса BlueSphere2

Пример (для 5 (5im) версии ТПО):

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!--Создание экрана ввода номера брони-->
 <screen type="numeric" title ="Введите номер брони" id="0screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 6. Название поля
ввода: "Номер брони"-->
   <text-field id="id1" title="Номер брони" max-len="6"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules> <rule regex="^.{6}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите номер брони]]> </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <goto target="1screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="previous"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
 <!--Создание экрана ввода данных с несколькими полями-->
 <screen type="group" title ="Введите данные"
         id="1screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура с возможностью переключения языка,
максимальное количество вводимых символов 60. Название поля ввода: "ФИО
плательщика"-->
   <text-field id="fio" title="ФИО плательщика" max-len="60"
```

```
                keyboard="any:ru:en:symb:upper:true">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules>
      <rule regex="^[a-яA-Яёё-]{2,20}[\s]{1}[a-яA-Яёё-]{2,20}[\s]{1,3}
                  [a-яA-ЯЁё-]{2,20}([\s]{1}[a-яA-ЯЁё-]{2,20})?$"/>
     </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура с возможностью переключения языка,
максимальное количество вводимых символов 254. Название поля ввода:
"Адрес плательщика"-->
   <text-field id="address" title="Адрес плательщика" max-len="254"
                keyboard="any:ru:en:symb:upper:true">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules> <rule regex="^.{3,254}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
   </text-field>
   <!--Создание поля выбора типа оплаты-->
   <selector-field id="type" title="Тип оплаты">
    <items type="static">
     <item value="21" title="Паспорт РФ"/>
     <item value="22" title="Заграничный паспорт"/>
     <item value="31" title="Паспорт иностранного гражданина"/>
    </items>
   </selector-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <goto target="2screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнокпи "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="previous"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнокпи "Выход"-->
```

```
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
 <!--Создание группового экрана для ввода данных-->
 <screen type="group" title ="Введите данные"
         id="2screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 20. Название поля
ввода: "Серия документа"-->
   <text-field id="seria" title="Серия документа" max-len="20"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules> <rule regex="^\d{1,20}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите серию документа]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 20. Название поля
ввода: "Номер документа"-->
   <text-field id="nomer" title="Номер документа" max-len="13"
               keyboard="Digital">
   <!--Регулярное выражение для валидации вводимого номера-->
   <validator type="regex">
    <rules> <rule regex="^\d{1,13}$"/> </rules>
   </validator>
   <!--Подсказка, отображается на экране-->
   <help> <![CDATA[<html>Введите номер документа]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
буквенная и символьная клавиатура с возможностью переключения языка,
максимальное количество вводимых символов 254. Название поля ввода: "Кем
выдан"-->
   <text-field id="name" title="Кем выдан" max-len="254"
               keyboard="any:ru:en:symb:upper:true">
   <!--Регулярное выражение для валидации вводимых данных-->
   <validator type="regex">
    <rules> <rule regex="^.{1,254}$"/> </rules>
   </validator>
```

```
   <!--Подсказка, отображается на экране-->
   <help> <![CDATA[<html>Введите наименование органа,
          выдавшего документ]]></help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 8. Название поля
ввода: "Дата выдачи документа"-->
   <text-field id="date" title="Дата выдачи документа" max-len="8"
              keyboard="Digital">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules>
      <rule regex="^[0123]\d{1}(0[1-9]|1[0-2])(19|20)\d{2}$"/>
     </rules>
    </validator>
     <!--Подсказка, отображается на экране-->
     <help> <![CDATA[<html>Введите дату выдачи документа]]> </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <!--Модификация параметра date-->
    <modify src-key="date" t-value="^(\d{2})(\d{2})(\d{4})$"
           t-value-title="^(\d{2})(\d{2})(\d{4})$"
           v-value="$1.$2.$3" v-value-title="$1.$2.$3" />
    <goto target="confirm"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="1screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
 <!--Создание экрана подтверждения-->
 <screen type="confirm" decor="long"
         title ="Подтверждение данных платежа" id="confirm">
  <!--Список атрибутов, отображаемых на экране подтверждения-->
  <fields list="id1,fio,address,type,seria,nomer,name,date"/>
```

```
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <goto target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="2screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
</scenario>
```

4.6.11 ЭКРАН ПОДТВЕРЖДЕНИЯ С НАВИГАЦИЕЙ (CONFIRM/NAVI)

Экран с типом **`type=«confirm/navi»`** представляет собой экран подтверждения введенной ранее информации с элементами навигации (рисунок 4.6.11.1).

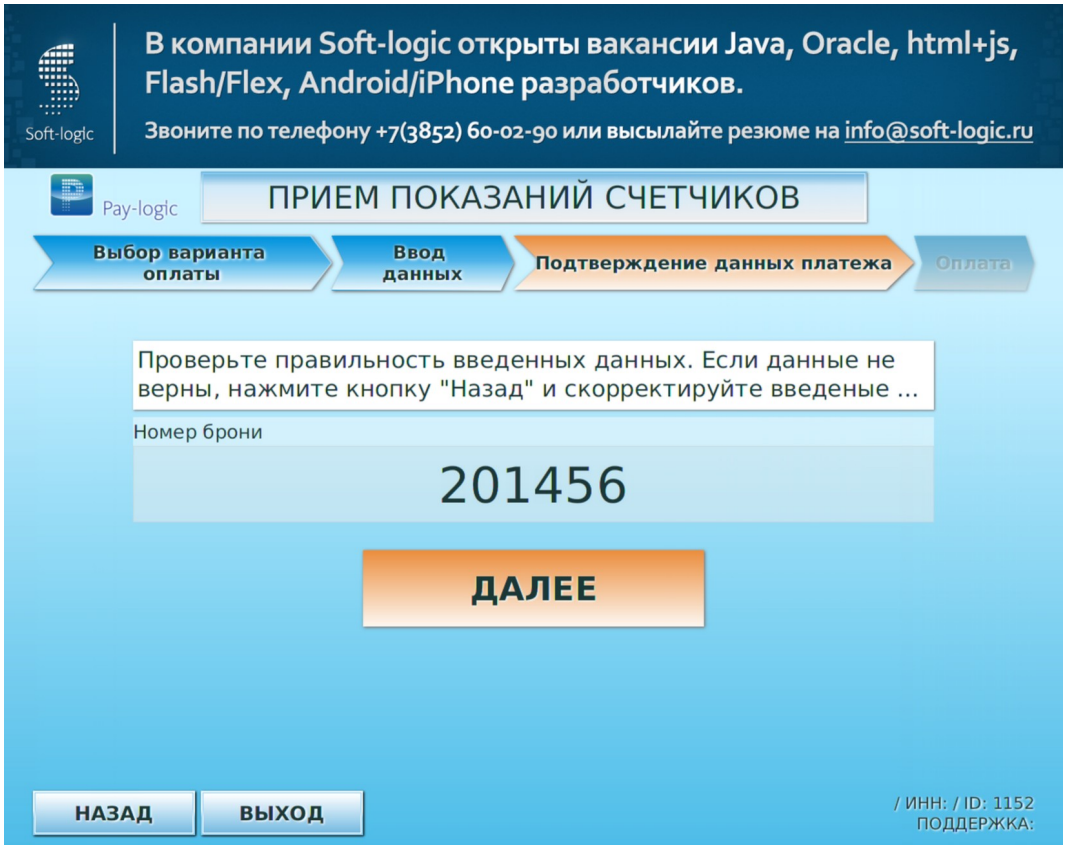


Рисунок 4.6.11.1 — Пример экрана CONFIRM/NAVI для интерфейса Blues

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!--Создание экрана для ввода числовой и текстовой информации-->
 <screen type="numeric" title ="Введите номер брони" id="0screen">
  <fields> ... </fields>
  <actions> ... </actions>
 </screen>
 <!--Создание экрана ввода данных с несколькими полями-->
 <screen type="group" title ="Введите данные" id="1screen">
  <fields> ... </fields>
  <actions> ... </actions>
 </screen>
 <!--Создание экрана ввода с несколькими полями с навигацией-->
 <screen type="group/navi" decor="list" title ="Введите данные"
         id="2screen">
  <fields> ... </fields>
  <navigation> ... </navigation>
  <actions> ... </actions>
 </screen>
 <!--Создание экрана подтверждения с навигацией-->
 <screen type="confirm/navi" decor="long"
        title ="Подтверждение данных платежа" id="confirm">
  <!--Список атрибутов, отображаемых на экране подтверждения-->
  <fields list="id1,fio,address,type,seria,nomer,name,date"/>
  <!--Создание навигации на экране-->
  <navigation>
   <!--Создание первой кнопки навигации-->
   <action type="navi1" title="Выбор варианта оплаты">
    <goto target="0screen"/>
   </action>
   <!--Создание второй кнопки навигации-->
   <action type="navi2" title="Ввод данных" >
    <if condition = "tip == 100">
     <then> <goto target="01screen"/> </then>
     <else> <goto target="02screen"/> </else>
    </if>
   </action>
   <!--Создание третьей кнопки навигации-->
   <action type="navi3" title="Подтверждение данных платежа"
           current="true">
    <goto target="confirm"/>
   </action>
```

```
   <!--Создание четвертой кнопки навигации-->
   <action type="navi4" title="Оплата">
    <goto target="pay"/>
   </action>
  </navigation>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <goto-action type="Next" title="Далее" target="pay"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <goto-action type="Prev" title="Назад" target="2screen"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
</scenario>
```

4.6.12 ЭКРАН ПОДТВЕРЖДЕНИЯ С НЕСКОЛЬКИМИ ПОЛЯМИ (CONFIRM/MULTIPLE)

Для отображения нескольких полей на экране подтверждения необходимо использовать экран с типом **`type=«confirm/multiple»`** (рисунок 4.6.12.1).

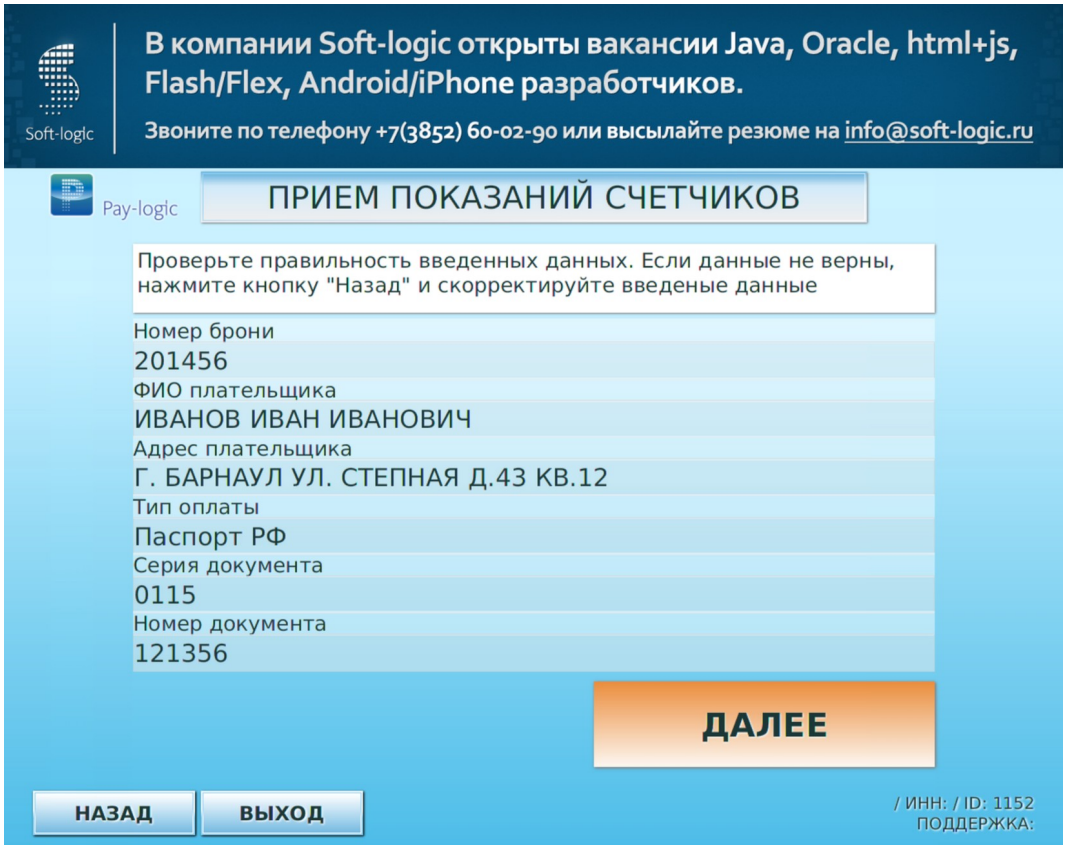


Рисунок 4.6.12.1 — Пример экрана CONFIRM/MULTIPLE для интерфейса Blues

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
<!--Создание экрана ввода числовой и текстовой информации-->
<screen type="numeric" title ="Введите номер брони" id="0screen">
 <fields>
  <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 6. Название поля
ввода: "Номер брони"-->
  <text-field id="id1" title="Номер брони" max-len="6"
              keyboard="Digital">
   <!--Регулярное выражение для валидации вводимого номера-->
   <validator type="regex">
    <rules> <rule regex="^.{6}$"/> </rules>
   </validator>
   <!--Подсказка, отображается на экране-->
   <help> <![CDATA[<html>Введите номер брони]]> </help>
  </text-field>
 </fields>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
  <goto-action type="Next" title="Далее" target="1screen"/>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
  <goto-action type="Prev" title="Назад" target="previous"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
  <goto-action type="Exit" title="Выход" target="exit"/>
   </actions>
</screen>
<!--Создание экрана ввода данных с несколькими полями-→
<screen type="group" title="Введите данные" id="1screen">
 <fields>
  <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода - русский, без возможности
смены, максимальное количество вводимых символов 60. Название поля ввода:
"ФИО плательщика"-->
  <text-field id="fio" title="ФИО плательщика" max-len="60"
              keyboard="any:[ru,symb]:upper:true">
   <!--Регулярное выражение для валидации вводимых данных-->
   <validator type="regex">
    <rules>
     <rule regex="^[a-яА-Яёё-]{2,20}[\s]{1}[a-яА-Яёё-]{2,20}[\s]
                 {1,3}[a-яА-ЯЁё-]{2,20}([\s]{1}[a-яА-ЯЁё-]{2,20})?$"/>
   </rules>
```

```
  </validator>
  <!--Подсказка, отображается на экране-->
  <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
 </text-field>
 <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода - русский, без возможности
смены, максимальное количество вводимых символов 60. Название поля ввода:
"Адрес плательщика"-->
 <text-field id="address" title="Адрес плательщика" max-len="254"
             keyboard="any:[ru,symb]:upper:true">
  <!--Регулярное выражение для валидации вводимых данных-->
  <validator type="regex">
   <rules> <rule regex="^.{3,254}$"/> </rules>
  </validator>
  <!--Подсказка, отображается на экране-->
  <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
 </text-field>
 <!--Создание поля выбора типа оплаты-->
 <selector-field id="type" title="Тип оплаты">
  <items type="static">
   <item value="21" title="Паспорт РФ"/>
   <item value="22" title="Заграничный паспорт"/>
   <item value="31" title="Паспорт иностранного гражданина"/>
  </items>
 </selector-field>
</fields>
<actions>
 <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
 <goto-action type="Next" title="Далее" target="2screen"/>
 <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
 <goto-action type="Prev" title="Назад" target="previous"/>
 <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
 <goto-action type="Exit" title="Выход" target="exit"/>
</actions>
</screen>
<!--Создание экрана ввода с несколькими полями и навигацией-->
<screen type="group/navi" decor="list" title ="Введите данные"
        id="2screen">
 <fields>
  <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 20. Название поля
ввода: "Серия документа"-->
  <text-field id="seria" title="Серия документа" max-len="20"
```

```
              keyboard="Digital">
  <!--Регулярное выражение для валидации вводимого номера-->
  <validator type="regex">
   <rules> <rule regex="^\d{1,20}$"/> </rules>
  </validator>
  <!--Подсказка, отображается на экране-->
  <help> <![CDATA[<html>Введите серию документа]]> </help>
 </text-field>
 <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 13. Название поля
ввода: "Номер документа"-->
 <text-field id="nomer" title="Номер документа" max-len="13"
              keyboard="Digital">
  <!--Регулярное выражение для валидации вводимого номера-->
  <validator type="regex">
   <rules> <rule regex="^\d{1,13}$"/> </rules>
  </validator>
  <!--Подсказка, отображается на экране-->
  <help> <![CDATA[<html>Введите номер документа]]> </help>
 </text-field>
 <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский, без возможности
смены, максимальное количество вводимых символов 254. Название поля
ввода: "Кем выдан"-->
 <text-field id="name" title="Кем выдан" max-len="254"
              keyboard="any:[ru,symb]:upper:true">
  <!--Регулярное выражение для валидации вводимых данных-->
  <validator type="regex">
   <rules> <rule regex="^.{1,254}$"/> </rules>
  </validator>
  <!--Подсказка, отображается на экране-->
  <help>
   <![CDATA[<html>Введите наименование органа, выдавшего документ]]>
  </help>
 </text-field>
 <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 8. Название поля
ввода: "Дата выдачи документа"-->
 <text-field id="date" title="Дата выдачи документа" max-len="8"
             keyboard="Digital">
  <!--Регулярное выражение для валидации вводимых данных-->
  <validator type="regex">
   <rules>
```

```
  <rule regex="^[0123]\d{1}(0[1-9]|1[0-2])(19|20)\d{2}$"/>
 </rules>
</validator>
 <!--Подсказка, отображается на экране-->
 <help> <![CDATA[<html>Введите дату выдачи документа]]> </help>
</text-field>
</fields>
<!--Создание навигации на экране-->
<navigation>
<!--Создание первой кнопки навигации-->
<action type="navi1" title="Номер брони">
 <goto target="0screen"/>
</action>
<!--Создание второй кнопки навигации-->
<action type="navi2" title="Данные плательщика">
 <goto target="1screen"/>
</action>
<!--Создание третьей кнопки навигации-->
<action type="navi3" title="Паспортные данные" current="true">
 <goto target="2screen"/>
</action>
</navigation>
<actions>
<!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
<action type="Next" title="Далее">
 <!--Модификация параметра date-->
 <modify src-key="date" t-value="^(\d{2})(\d{2})(\d{4})$"
        t-value-title="^(\d{2})(\d{2})(\d{4})$"
        v-value="$1.$2.$3" v-value-title="$1.$2.$3" />
 <goto target="confirm"/>
</action>
<!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
<goto-action type="Prev" title="Назад" target="1screen"/>
<!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
<goto-action type="Exit" title="Выход" target="exit"/>
</actions>
</screen>
<!--Создание экрана подтверждения-->
<screen type="confirm/multiple" decor="long"
        title ="Подтверждение данных платежа" id="confirm">
<!--Список атрибутов, отображаемых на экране подтверждения-->
<fields list="id1,fio,address,type,seria,nomer,name,date"/>
<actions>
```

```
 <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
 <goto-action type="Next" title="Далее" target="pay"/>
 <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
 <goto-action type="Prev" title="Назад" target="2screen"/>
 <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
 <goto-action type="Exit" title="Выход" target="exit"/>
 </actions>
</screen>
</scenario>
```

4.6.13 ЭКРАН ПОДТВЕРЖДЕНИЯ С НЕСКОЛЬКИМИ ПОЛЯМИ И НАВИГАЦИЕЙ (CONFIRM/MULTIPLE/NAVI)

Для отображения нескольких полей на экране подтверждения с навигацией необходимо использовать экран с типом **`type=«confirm/multiple/navi»`**.

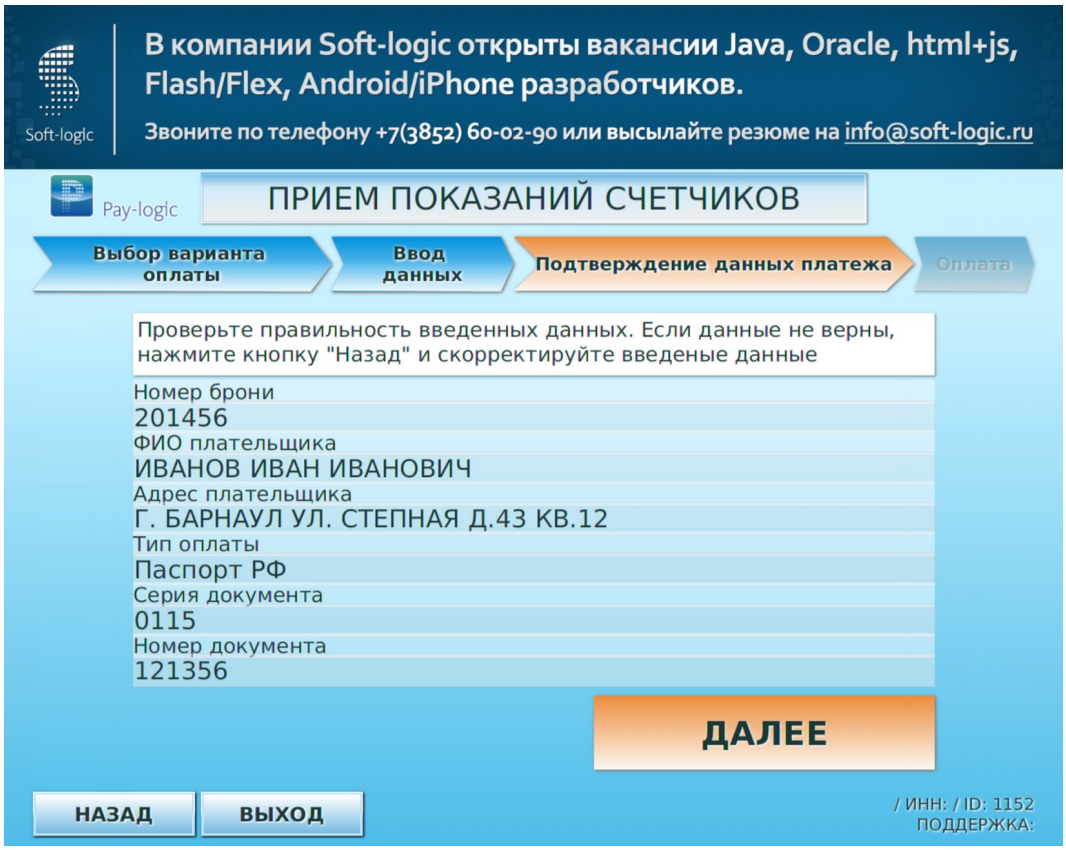


Рисунок 4.6.13.1 — Пример экрана CONFIRM/MULTIPLE/NAVI для интерфейса Blues

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!--Создание экрана ввода числовой и текстовой информации-->
 <screen type="numeric" title ="Введите номер брони" id="0screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 6. Название поля
ввода: "Номер брони"-->
   <text-field id="id1" title="Номер брони" max-len="6"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules> <rule regex="^.{6}$"/> </rules>
    </validator>
    <help> <![CDATA[<html>Введите номер брони]]> </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <goto-action type="Next" title="Далее" target="1screen"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <goto-action type="Prev" title="Назад" target="previous"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
 <!--Создание экрана ввода данных с несколькими полями-->
 <screen type="group" title ="Введите данные" id="1screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский, без возможности
смены, максимальное количество вводимых символов 60. Название поля ввода:
"ФИО плательщика"-->
   <text-field id="fio" title="ФИО плательщика" max-len="60"
               keyboard="any:[ru,symb]:upper:true">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules>
      <rule regex="^[а-яА-Яёё-]{2,20}[\s]{1}[а-яА-Яёё-]
                  {2,20}[\s]{1,3}[а-яА-Яёё-]{2,20}([\s]{1}
                  [а-яА-ЯЁё-]{2,20})?$"/>
     </rules>
```

```
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода - русский, без возможности
смены, максимальное количество вводимых символов 254. Название поля
ввода: "Адрес плательщика"-->
   <text-field id="address" title="Адрес плательщика" max-len="254"
               keyboard="any:[ru,symb]:upper:true">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules> <rule regex="^.{3,254}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
   </text-field>
   <!--Создание поля выбора типа оплаты-->
   <selector-field id="type" title="Тип оплаты">
    <items type="static">
     <item value="21" title="Паспорт РФ"/>
     <item value="22" title="Заграничный паспорт"/>
     <item value="31" title="Паспорт иностранного гражданина"/>
    </items>
   </selector-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <goto-action type="Next" title="Далее" target="2screen"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <goto-action type="Prev" title="Назад" target="previous"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
 <!--Создание экрана ввода данных с несколькими полями с навигацией-->
 <screen type="group/navi" decor="list" title ="Введите данные"
         id="2screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 20. Название поля
ввода: "Серия документа"-->
   <text-field id="seria" title="Серия документа" max-len="20"
```

```
                keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules> <rule regex="^\d{1,20}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите серию документа]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 13. Название поля
ввода: "Номер документа"-->
   <text-field id="nomer" title="Номер документа" max-len="13"
                keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules> <rule regex="^\d{1,13}$"/> </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите номер документа]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский, без возможности
смены, максимальное количество вводимых символов 254. Название поля
ввода: "Кем выдан"-->
   <text-field id="name" title="Кем выдан" max-len="254"
                keyboard="any:[ru,symb]:upper:true">
   <!--Регулярное выражение для валидации вводимых данных-->
   <validator type="regex">
    <rules> <rule regex="^.{1,254}$"/> </rules>
   </validator>
   <!--Подсказка, отображается на экране-->
   <help>
   <![CDATA[<html>Введите наименование органа, выдавшего
            документ]]></help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 8. Название поля
ввода: "Дата выдачи документа"-->
   <text-field id="date" title="Дата выдачи документа" max-len="8"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules>
```

```
     <rule regex="^[0123]\d{1}(0[1-9]|1[0-2])(19|20)\d{2}$"/>
    </rules>
   </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите дату выдачи документа]]> </help>
  </text-field>
 </fields>
 <!--Создание навигации на экране-->
 <navigation>
  <action type="navi1" title="Номер брони">
   <goto target="0screen"/>
  </action>
  <!--Создание первой кнопки навигации-->
  <action type="navi2" title="Данные плательщика">
   <goto target="1screen"/>
  </action>
  <!--Создание третьей кнопки навигации-->
  <action type="navi3" title="Паспортные данные" current="true">
   <goto target="2screen"/>
  </action>
 </navigation>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
  <action type="Next" title="Далее">
   <!--Модификация параметра date-->
   <modify src-key="date" t-value="^(\d{2})(\d{2})(\d{4})$"
           t-value-title="^(\d{2})(\d{2})(\d{4})$"
           v-value="$1.$2.$3" v-value-title="$1.$2.$3" />
   <goto target="confirm"/>
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
  <goto-action type="Prev" title="Назад" target="1screen"/>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
  <goto-action type="Exit" title="Выход" target="exit"/>
 </actions>
</screen>
<!--Создание экрана подтверждения с навигацией-->
<screen type="confirm/multiple/navi" decor="long"
        title ="Подтверждение данных платежа" id="confirm">
 <!--Список атрибутов, отображаемых на экране подтверждения-->
 <fields list="id1,fio,address,type,seria,nomer,name,date"/>
 <!--Создание навигации на экране-->
 <navigation>
```

```
  <!--Создание первой кнопки навигации-→
  <action type="navi1" title="Выбор варианта оплаты">
   <goto target="0screen"/>
  </action>
  <!--Создание второй кнопки навигации-->
  <action type="navi2" title="Ввод данных" >
   <!--Если "tip == 100", то переход на экран с id 01screen-->
   <if condition = "tip == 100">
    <then> <goto target="01screen"/> </then>
    <!--Иначе - переход на экран с id 02screen-->
    <else> <goto target="02screen"/> </else>
   </if>
  </action>
  <!--Создание третьей кнопки навигации-->
  <action type="navi3" title="Подтверждение данных платежа"
          current="true">
   <goto target="confirm"/>
  </action>
  <!--Создание четвертой кнопки навигации-->
  <action type="navi4" title="Оплата">
   <goto target="pay"/>
  </action>
 </navigation>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
  <goto-action type="Next" title="Далее" target="pay"/>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-→
  <goto-action type="Prev" title="Назад" target="2screen"/>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
  <goto-action type="Exit" title="Выход" target="exit"/>
 </actions>
</screen>
</scenario>
```

4.6.14 ЭКРАН ВЫБОРА ДАТЫ (DATE)

Экран с типом **`type=«date»`** предназначен для выбора даты (рисунки 4.6.14.1, 4.6.14.2). Для данного типа экрана доступно поле ввода `<date-field>` (раздел 5.11).

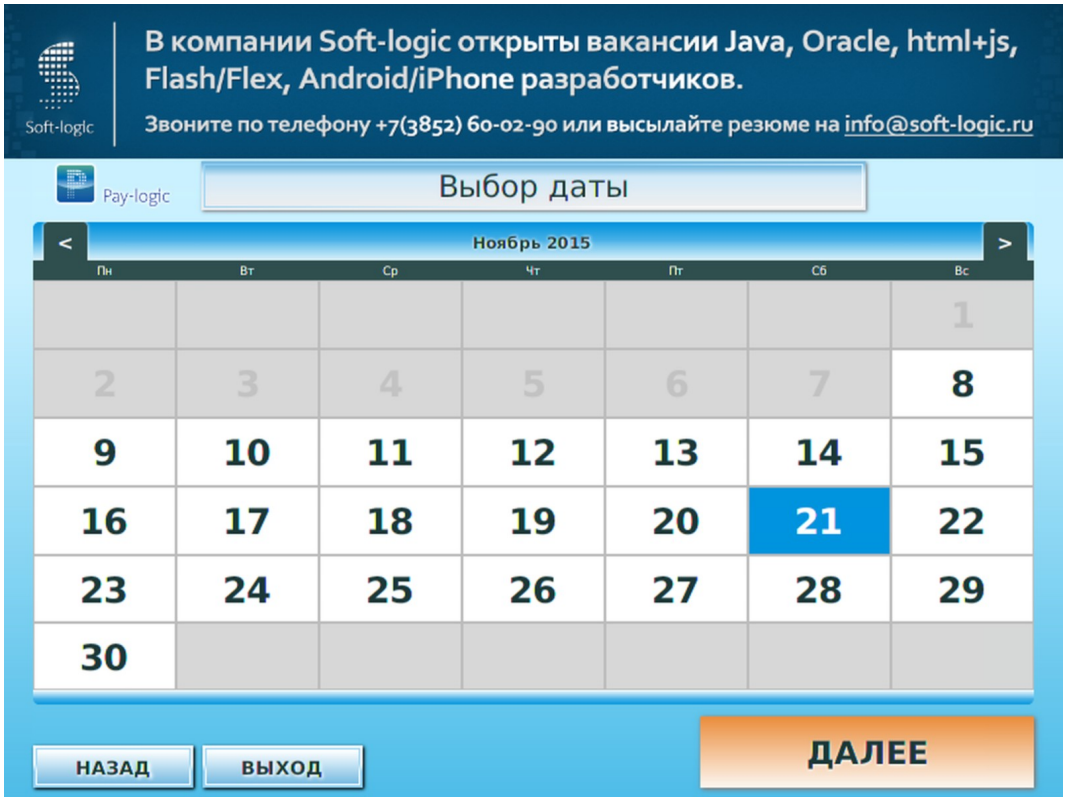


Рисунок 4.6.14.1 — Пример экрана DATE для интерфейса Blues

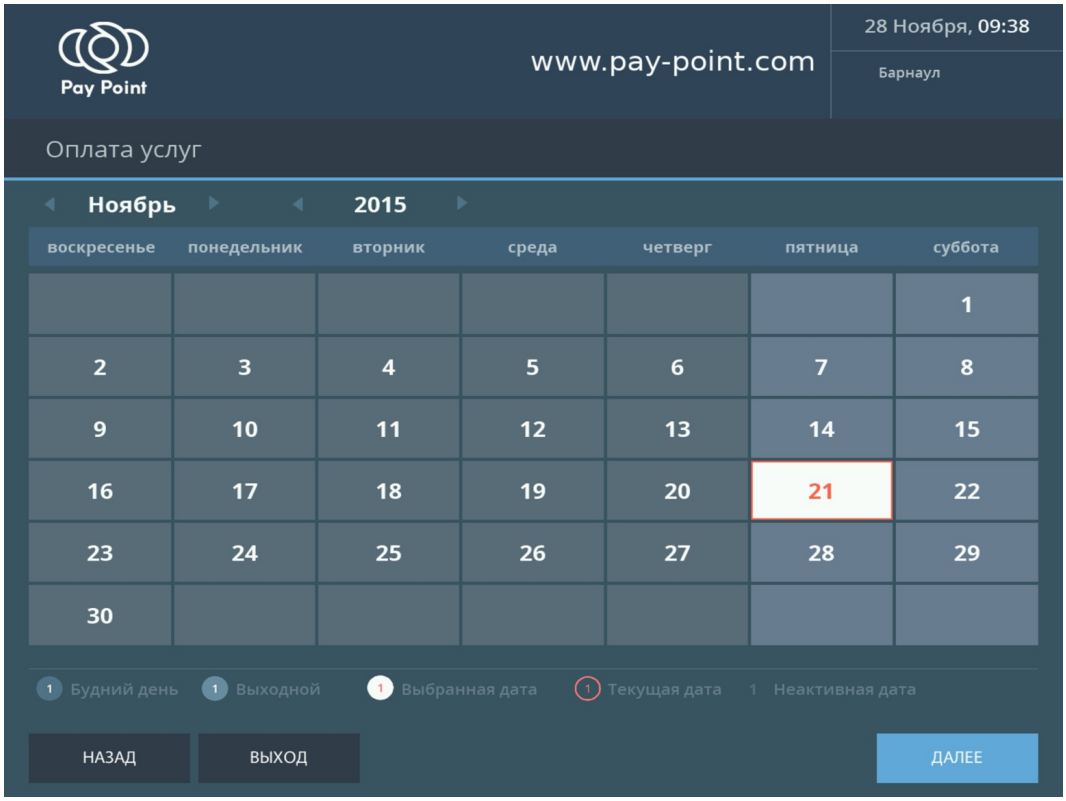


Рисунок 4.6.14.2 — Пример экрана DATE для интерфейса Smoke

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="input1">
 <!--Создание экрана выбора даты-->
 <screen type="date" title ="Экран выбора даты" id="input1">
  <fields>
   <!--Создание поля выбора даты-->
   <date-field id="date" title="Выбор даты" format="dd.MM.yyyy"
               format-title="dd.MM.yyyy" from="01.01.2010"
               to="01.01.2020" default="21.11.2015" />
  </fields>
  <actions> ... </actions>
 </screen>
</scenario>
```

4.6.15 ВСПЛЫВАЮЩИЙ ЭКРАН ДАТЫ (DATE/POPUP)

Экран с типом **`type=«date/popup»`** предназначен для выбора даты, но отображается не на весь экран терминала, как «DATE», а в виде всплывающего окна. Для данного типа экрана доступно поле ввода `<date-field>` (раздел 5.11).

Рисунок 4.6.15.1 — Экран DATE/POPUP

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="input1">
 <!--Создание всплывающего экрана выбора даты-->
 <screen type="date/popup" title ="Экран даты" id="input1">
  <fields>
   <!--Создание поля выбора даты-->
   <date-field id="date" title="Выбор даты" format="dd.MM.yyyy"
               format-title="dd.MM.yyyy" from="01.01.2010"
               to="01.01.2020" default="21.11.2015" />
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее"> <goto target="pay"/> </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад"> <goto target="previous"/> </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход"> <goto target="exit"/> </action>
  </actions>
 </screen>
</scenario>
```

4.6.16 ЭКРАН ВВОДА ЧИСЛОВОЙ ИНФОРМАЦИИ В 5 ВЕРСИИ ТПО (DIGITAL)

Экран с типом **`type=«Digital»`** — экран ввода номера (только большая цифровая клавиатура), предназначен для ввода числовой информации, рисунок 4.6.16.1. Используется в ТПО 5 и 7 версий. Аналог экрана «NUMERIC».

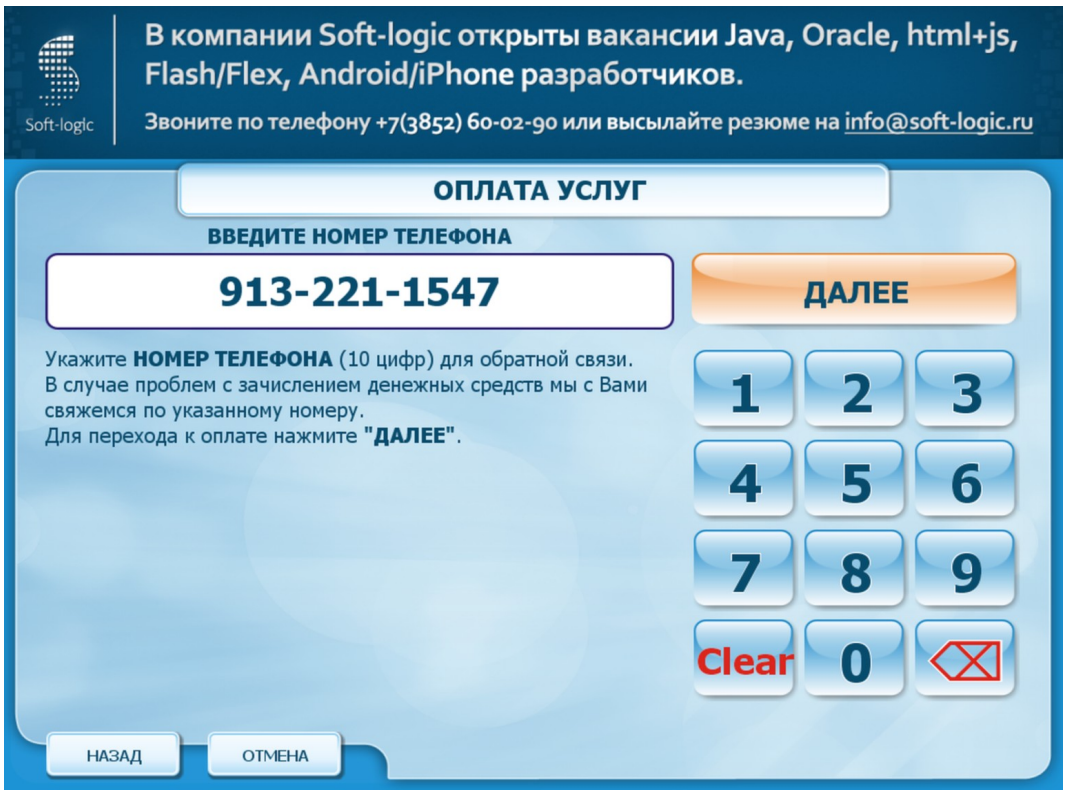


Рисунок 4.6.16.1 — Пример экрана DIGITAL для интерфейса BlueSphere2

Пример:

```
<!--Создание экрана ввода числовой и текстовой информации-->
<screen type="Digital" decor="simple" id="input2"
        title="Введите номер телефона">
 <fields>
  <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 10. Название поля
ввода: "Телефон"-->
  <text-field id="id1" title="Телефон" max-len="10" keyboard="Digital">
   <!--Регулярное выражение для валидации вводимого номера-->
   <validator type="regex">
    <rules> <rule regex="^\d{10}$"/> </rules>
   </validator>
   <!--Регулярное выражение для форматирования вводимого номера на
экране-->
   <formatter type="regex">
    <rules default="8(***)***-**-**"/>
   </formatter>
   <!--Подсказка, отображается на экране-->
   <help>
    <![CDATA[<html>
       Укажите <font weight="bold">НОМЕР ТЕЛЕФОНА</font> (10 цифр)
       для обратной связи<br>
       В случае проблем с зачисление денежных средств мы с Вами<br>
       свяжемся по указанному номеру<br>
       Для перехода к оплате нажмите
       <font weight="bold">”ДАЛЕЕ”</font>.]]>
   </help>
  </text-field>
 </fields>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
  <action type="Next" title="Далее">
   <!--Модификация параметра id1-->
   <modify src-key="id1" t-key="^(.*)$" v-key="phone"
           t-value="^(\d{10}).*$" v-value="$1"
           t-value-title="^(\d{10}).*$"
           v-value-title="$1" t-key-title="^(.*)$"
           v-key-title="Номер телефона"/>
   <goto target="input3"/>
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
```

```
  <action type="Prev" title="Назад">
   <goto target="input1"/>
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
  <action type="Exit" title="Выход">
   <goto target="exit"/>
  </action>
 </actions>
</screen>
```

4.6.17 ЭКРАН ВВОДА ЧИСЛОВОЙ ИНФОРМАЦИИ С ОТОБРАЖЕНИЕМ СТАВКИ КОМИССИИ (DIGITAL/COMMISSION)

Экран **`type=«Digital/commission»`** предназначен для ввода числовой и текстовой информации с отображением информации о комиссии.

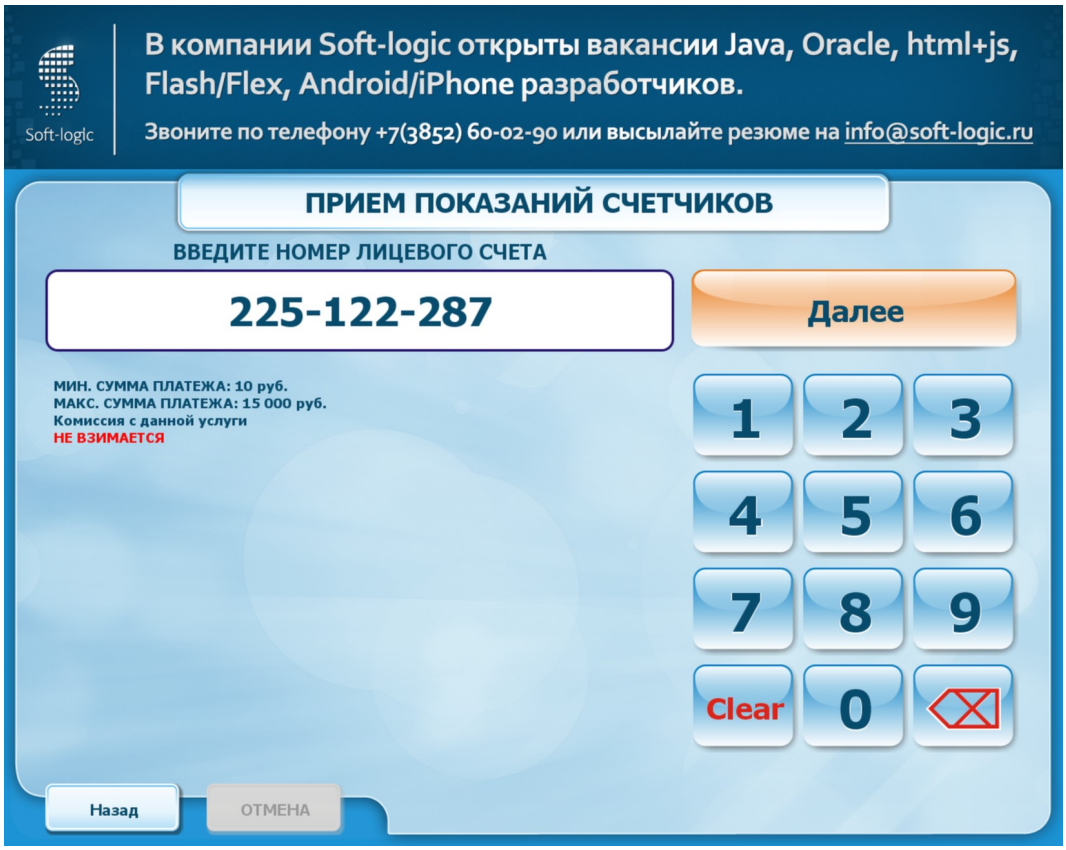


Рисунок 4.6.17.1 — Пример экрана DIGITAL/COMMISSION (BlueSphere2)

Пример (для ТПО 5 версии):

```
<!--Создание экрана ввода числовой и текстовой информации-->
<screen type="Digital" decor="commission"
        title="Введите номер лицевого счета" id="screen_2">
 <fields>
  <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 9. Название поля
ввода: "Номер лицевого счета"-->
  <text-field id="id1" title="Номер лицевого счета"
              max-len="9" keyboard="Digital">
   <!--Регулярное выражение для валидации вводимого номера-->
   <validator type="regex">
    <rules> <rule regex="^\d{9}$"/> </rules>
   </validator>
   <!--Регулярное выражение для форматирования вводимого номера на
экране-->
   <formatter type="regex">
    <rules default="***-***-***"/>
   </formatter>
  </text-field>
 </fields>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
  <action type="Next" title="Далее">
   <goto target="confirm"/>
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
  <action type="Prev" title="Назад">
   <goto target="screen_1"/>
  </action>
 </actions>
</screen>
```

4.6.18 ГРУППОВОЙ ЭКРАН (GROUP)

Экран с типом «GROUP» позволяет группировать поля ввода различных типов, рисунки 4.6.18.1, 4.6.18.2. Для группового экрана дополнительно доступен атрибут **`focused-field`** — содержит название поля ввода. Для экрана доступны следующие поля ввода `<text-field>` (раздел 5.7), `<numeric-field>` (раздел 5.8), `<checkbox-field>` (раздел 5.9), `<selector-field>` (раздел 5.10.1), `<date-field>` (раздел 5.11), `<autocomplete-field>` (раздел 5.12).

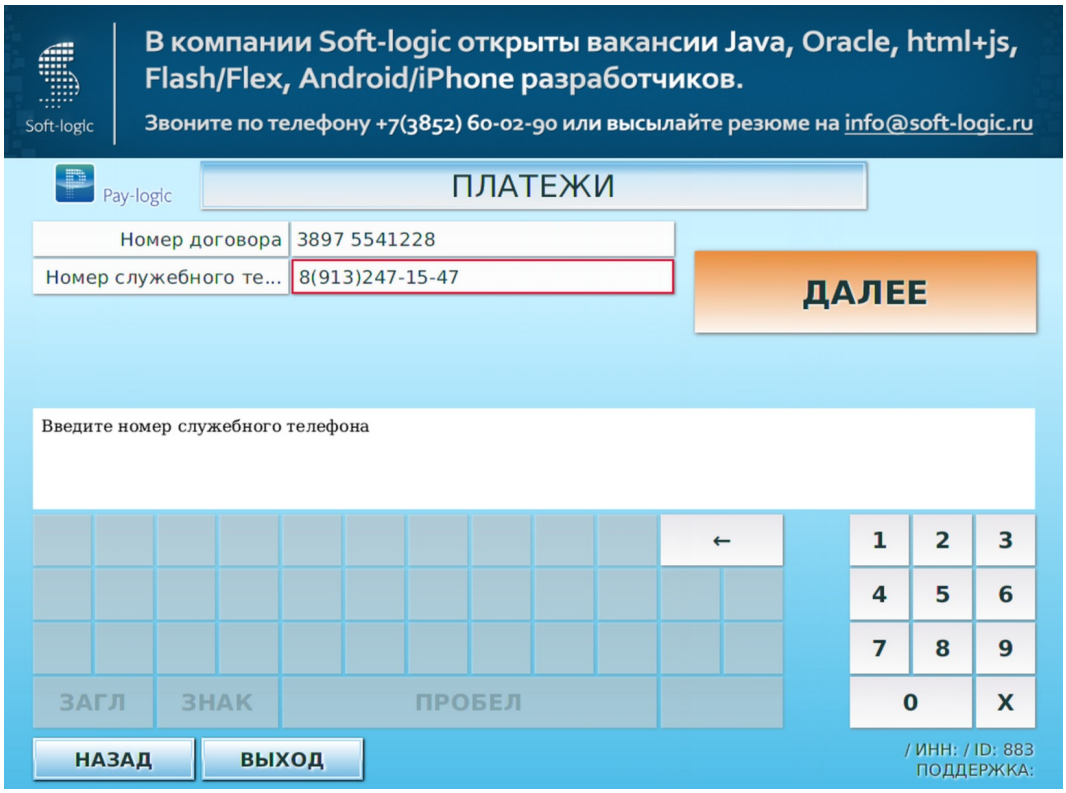


Рисунок 4.6.18.1 — Пример экрана GROUP для интерфейса Blues

Рекомендуется размещать на групповом экране не более 4 элементов ввода. При большем количестве элементов на экране появляется вертикальная полоса прокрутки,

которую клиент по каким-либо причинам может не заметить, вследствие чего не заполнить ряд полей.

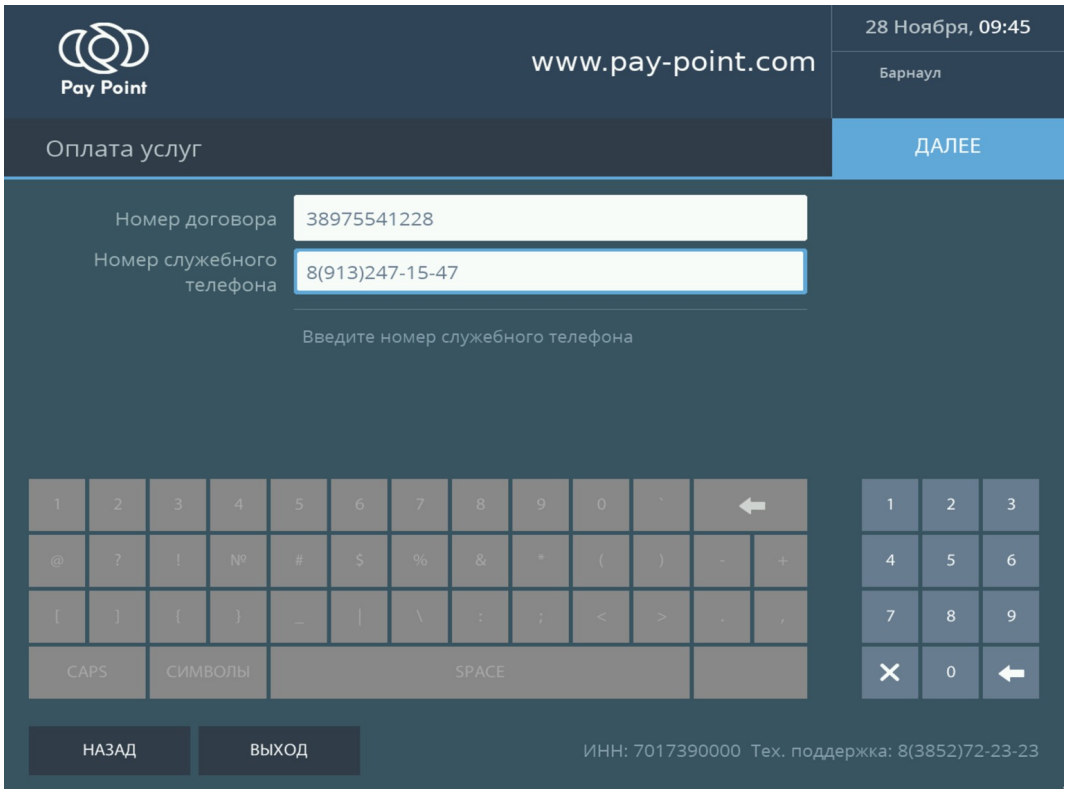


Рисунок 4.6.18.2 — Пример экрана GROUP для интерфейса Smoke

Пример:

```
<!--Создание экрана ввода данных с несколькими полями-->
<screen type="group" title ="Введите данные" id="2screen">
<fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 11. Название поля
ввода: "Номер договора"-->
   <text-field id="seria" title="Номер договора" max-len="11"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules>
       <rule regex="^\d{1,11}$"/>
     </rules>
```

```
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите номер документа]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 13. Название поля
ввода: "Номер служебного телефона"-->
   <text-field id="nomer" title="Номер служебного телефона" max-len="13"
               keyboard="Digital">
      <!--Регулярное выражение для валидации вводимого номера-->
      <validator type="regex">
     <rules> <rule regex="^\d{10}$"/> </rules>
    </validator>
     <!--Регулярное выражение для форматирования вводимого номера на
экране-->
    <formatter type="regex">
     <rules default="8(***)***-**-**"/>
    </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите номер служебного телефона </help>
   </text-field>
  </fields>
  <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <!--Модификация параметра date-->
    <modify src-key="date" t-value="^(\d{2})(\d{2})(\d{4})$"
            t-value-title="^(\d{2})(\d{2})(\d{4})$"
            v-value="$1.$2.$3" v-value-title="$1.$2.$3" />
    <goto target="confirm"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <goto-action type="Prev" title="Назад" target="1screen"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
```

4.6.19 ЭКРАН ВВОДА СУММЫ И ПОКАЗАНИЙ (GROUP/METER)

Экран с типом «GROUP/METER» используется для ввода текущих показаний счетчиков и суммы оплаты (рисунок 4.6.19.1). На данном экране сначала отображаются поля для ввода информации в секции `<fields>`. Затем введенные данные упаковываются в элемент типа data (подробно о корзине платежей в разделе 7).

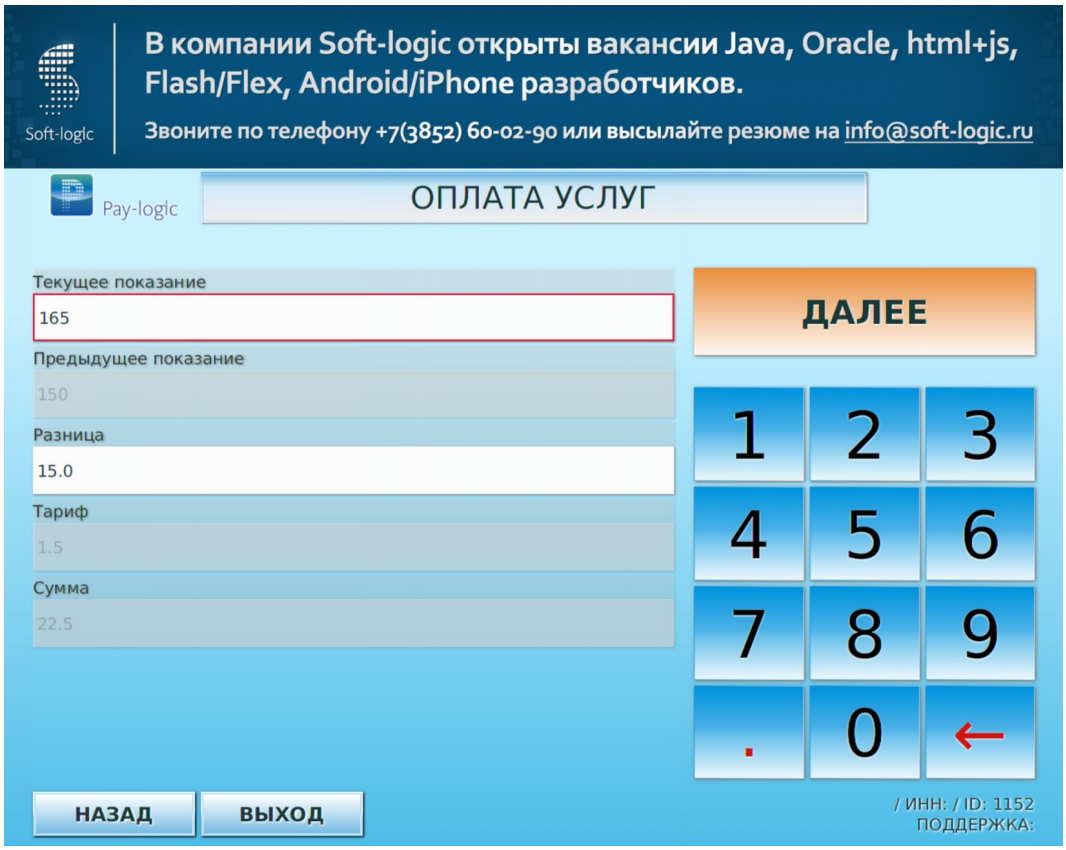


Рисунок 4.6.19.1 — Пример экрана GROUP/METER для интерфейса Blues

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="init_screen">
 <!--Особенности проведения группового платежа-->
 <payment-params key="#services" distribution-type="default"
                 single-check="true" />
 <!--Создание экрана формирования оффлайн данных-->
 <screen type="Void" decor="simple" title ="" id="init_screen">
  <fields/>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <!-- Ид услуги, обязателен-->
    <set key="#id" value="s1"/>
    <!-- Название услуги, обязателено-->
    <set key="name" value="Вода горячая"/>
    <!-- Текущее показание: null или "" -->
    <set key="current" value=""/>
    <!-- Предыдущее показание: null или "" или число > 0-->
    <set key="previous" value="" />
    <!-- Тариф, обязателен →
    <set key="tariff" value="2.5"/>
    <!-- Разница: null или "" -->
    <set key="consumption" value=""/>
    <!-- Сумма: null или "" -->
    <set key="summ" value=""/>
    <!-- Всегда должно присутствовать и равно 1 -->
    <set key="selected" value="1"/>
    <!--Упаковка набора элементов-->
    <pack type="data" key="dt1" elements="#id,name,current,previous,
                                    tariff,consumption,selected,summ"/>
    <set key="#id" value="s2"/>
    <set key="name" value="Вода холодная"/>
    <set key="current" value=""/>
    <set key="previous" value="150"/>
    <set key="tariff" value="1.5" />
    <set key="consumption" value="" />
    <set key="summ" value=""/>
    <set key="selected" value="1"/>
    <!--Упаковка набора элементов-->
    <pack type="data" key="dt2" elements="#id,name,current,previous,
                                    tariff,consumption,selected,summ"/>
    <set key="#id" value="s3"/>
```

```
    <set key="name" value="Электроэнергия"/>
    <set key="current" value=""/>
    <set key="previous" value="50"/>
    <set key="tariff" value="5.25"/>
    <set key="consumption" value=""/>
    <set key="summ" value=""/>
    <set key="selected" value="1"/>
    <!--Упаковка набора элементов-->
    <pack type="data" key="dt3" elements="#id,name,current,previous,
                                    tariff,consumption,selected,summ"/>
    <!-- Ид услуги, обязателен-->
    <set key="#id" value="s4"/>
    <!-- Название услуги, обязателено-->
    <set key="name" value="Водоотведение"/>
    <!-- Текущее показание: null или "" -->
    <set key="current" value=""/>
    <!-- Предыдущее показание: null или "" -->
    <set key="previous" value=""/>
    <!-- Тариф, обязателен -->
    <set key="tariff" value="1.1"/>
    <!-- Разница: null или "" -->
    <set key="consumption" value=""/>
    <!-- Сумма: null или "" -->
    <set key="summ" value=""/>
    <!-- Всегда должно присутствовать и равно 1 →
    <set key="selected" value="1"/>
    <!-- Указатель на ИД услуг, по которым считать: через , ид услуг-->
    <set key="#ref" value="s1,s2"/>
    <!--Упаковка набора элементов-→
    <pack type="data" key="dt4" elements="#id,name,current,previous,
                                tariff,consumption,selected,summ,#ref"/>
    <!--Упаковка набора элементов-->
    <pack type="nested" key="#services" elements="dt1,dt2,dt3,dt4,"/>
    <!--Удаление переменных из контекста-->
    <clear keys="#id,name,current,previous,tariff,consumption,
                 selected,summ,#ref,dt1,dt2,dt3,dt4"/>
    <goto target="com_screen"/>
  </action>
 </actions>
</screen>
<!--Создание экрана выбора услуг-->
<screen type="Communal/meter" decor="simple"
        title="Выбор услуг" id="com_screen">
```

```
  <!--Отображение полей из объекта NestedData #services-->
  <fields key="#services"/>
   <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="Далее">
     <!--Получение итоговой суммы группового платежа-->
     <sum key="summ" result-key="ssum" input-data="#services"
          filter="selected=1"/>
     <!--Передача значения числовой переменной в сумму платежа без учета
комиссии-->
     <set-sum from="ssum" type="units"/>
     <!--Удаление переменных из контекста-->
     <clear keys="ssum"/>
     <goto target="pay"/>
   </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="exit"/>
   </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
   <action type="Edit" title="">
    <!--Распаковка набора элементов-->
    <unpack type="data" key="#selected"
            elements="current,previous,consumption,tariff,summ"/>
    <goto target="input_meter"/>
   </action>
  </actions>
 </screen>
  <!--Создание экрана ввода данных с несколькими полями-->
  <screen type="group/meter" decor="simple" title="Введите данные"
          id="input_meter">
   <fields>
    <!--Создание поля ввода числа в десятичном формате через точку.
Название поля ввода: "Текущее показание"-->
    <numeric-field id="current" title="Текущее показание" format="5.2">
     <!--Валидация вводимых данных-->
     <verify>
      <range begin="0.01" end="99999.99"/>
     </verify>
    </numeric-field>
```

```
    <!--Создание поля ввода числа в десятичном формате через точку.
Название поля ввода: "Предыдущее показание"-->
    <numeric-field id="previous" title="Предыдущее показание"
                   format="5.2">
     <!--Валидация вводимых данных-->
     <verify>
      <range begin="0.01" end="99999.99"/>
     </verify>
    </numeric-field>
    <!--Создание поля ввода числа в десятичном формате через точку.
Название поля ввода: "Разница"-->
    <numeric-field id="consumption" title="Разница" format="5.2">
     <!--Валидация вводимых данных-->
     <verify>
      <range begin="0.01" end="99999.99"/>
     </verify>
    </numeric-field>
    <!--Создание поля ввода числа в десятичном формате через точку.
Название поля ввода: "Тариф"-->
    <numeric-field id="tariff" title="Тариф" format="5.2">
     <!--Валидация вводимой суммы-->
     <verify>
      <range begin="0.01" end="99999.99"/>
     </verify>
    </numeric-field>
    <!--Создание поля ввода числа в десятичном формате через точку.
Название поля ввода: "Сумма"-->
    <numeric-field id="summ" title="Сумма" format="5.2">
     <!--Валидация вводимой суммы-->
     <verify>
      <range begin="0.01" end="99999.99"/>
     </verify>
    </numeric-field>
   </fields>
   <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="ДАЛЕЕ">
     <!--Упаковка набора элементов-->
     <pack type="data" key="#selected"
           elements="current,previous,consumption,tariff,summ"/>
     <!--Удаление переменных из контекста-->
     <clear keys="#selected,current,previous,consumption,tariff,summ"/>
     <goto target="com_screen"/>
```

```
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <action type="Prev" title="НАЗАД">
     <!--Удаление переменных из контекста-->
     <clear keys="#selected,current,previous,consumption,tariff,summ"/>
     <goto target="com_screen"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <action type="Exit" title="Выход">
     <goto target="exit"/>
    </action>
   </actions>
  </screen>
</scenario>
```

4.6.20 ГРУППОВОЙ ЭКРАН С НАВИГАЦИЕЙ (GROUP/NAVI)

Экран с типом **type=«group/navi»** представляет собой экран с навигационными кнопками, которые соответствуют шагам сценария. При помощи данных кнопок можно осуществить возврат назад на любой пройденный этап сценария или формы. Экраны с навигацией имеют такой же тип, как и обычные, только с постфиксом **«/navi»**. Доступен только в 7 версии ТПО. Навигация описана в разделе 4.4. Пример экрана приведен на рисунке 4.4.1.

4.6.21 ГРУППОВОЙ ЭКРАН ЧИСЛОВЫХ ПОЛЕЙ ВВОДА ДАННЫХ (GROUP/NUMERIC)

Экран с типом **`type=«group/numeric»`** используется для заполнения нескольких числовых полей на одном экране (рисунки 4.6.21.1, 4.6.21.2).

Рисунок 4.6.21.1 — Пример экрана GROUP/NUMERIC для интерфейса Blues

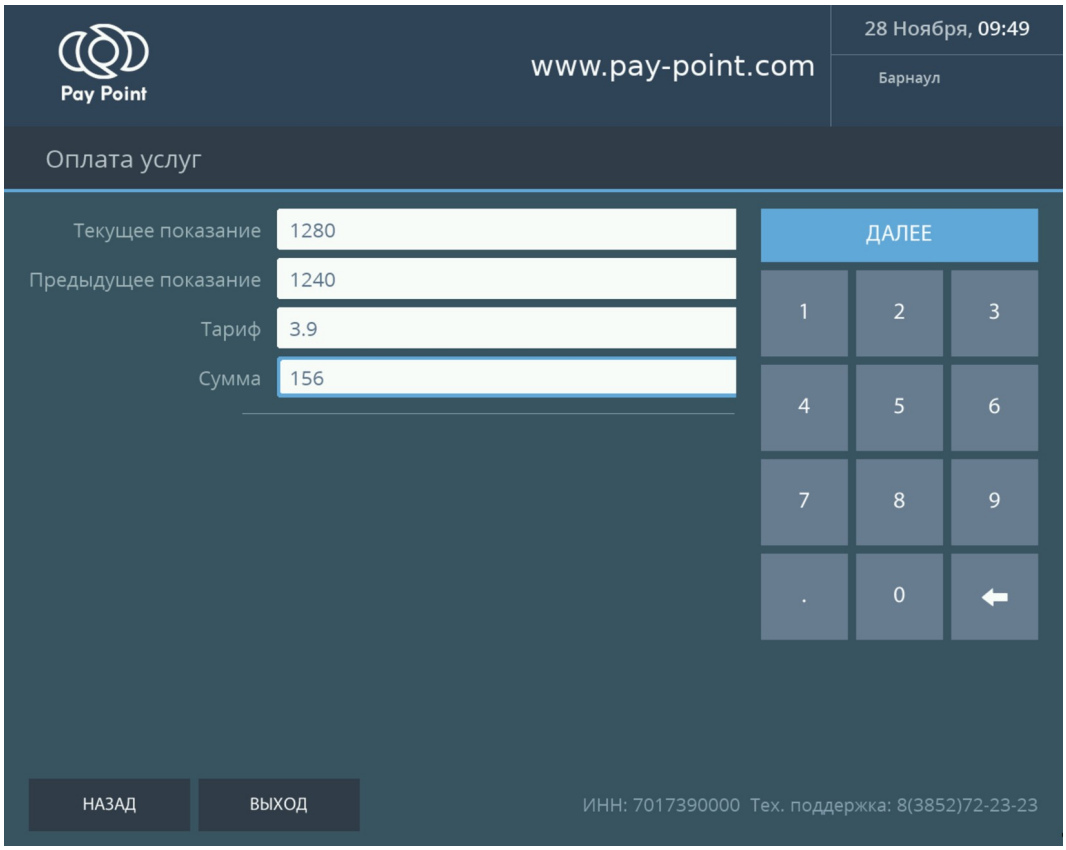


Рисунок 4.6.21.2 — Пример экрана GROUP/NUMERIC для интерфейса Smoke

Пример:

```
<!--Создание экрана ввода данных с несколькими полями-->
<screen type="group/numeric" title="Ввод данных" id="input_sum1">
 <fields>
  <!--Создание поля ввода числа в десятичном формате через точку.
Название поля ввода: "Текущее показание"-->
  <numeric-field id="current" title="Текущее показание" format="5.2">
   <!--Валидация вводимой суммы-->
   <verify>
    <range begin="0.00" end="99999.99" />
   </verify>
  </numeric-field>
  <!--Создание поля ввода числа в десятичном формате через точку.
Название поля ввода: "Предыдущее показание"-->
  <numeric-field id="previous" title="Предыдущее показание" format="5.2">
```

```
   <!--Валидация вводимых данных-->
   <verify>
    <range begin="0.01" end="99999.99" />
   </verify>
  </numeric-field>
  <!--Создание поля ввода числа в десятичном формате через точку.
Название поля ввода: "Разница"-->
  <numeric-field id="consumption" title="Разница" format="5.2">
   <!--Валидация вводимых данных-->
   <verify>
   <range begin="0.01" end="99999.99" />
   </verify>
  </numeric-field>
  <!--Создание поля ввода числа в десятичном формате через точку.
Название поля ввода: "Тариф"-->
  <numeric-field id="tariff" title="Тариф" format="5.2">
   <!--Валидация вводимой суммы-->
   <verify>
    <range begin="0.01" end="99999.99" />
   </verify>
  </numeric-field>
  <!--Создание поля ввода числа в десятичном формате через точку.
Название поля ввода: "Сумма"-->
  <numeric-field id="summ" title="Сумма" format="5.2">
   <!--Валидация вводимой суммы-→
   <verify>
    <range begin="0.01" end="99999.99" />
   </verify>
  </numeric-field>
 </fields>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
  <action type="Next" title="ДАЛЕЕ">
   <!--Если current эквивалентно ^\D*$-->
   <if condition="current ~ ^\D*$">
    <then>
     <!--Объявление переменной current-->
     <set key="current" key-title="Текущее показание" value="0"
          value-title="0" />
    </then>
    <else/>
   </if>
   <!--Объявление переменной selected-->
```

```
   <set key="selected" value="1" value-title="1" />
   <!--Упаковка набора элементов-->
   <pack type="data" key="#selected"
         elements="current,selected,consumption,previous,summ" />
   <!--Удаление переменных из контекста-->
   <clear keys="#selected,selected,name,consumption,
               current,previous,tariff,summ"/>
   <goto target="meters_screen" />
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
  <action type="Prev" title="НАЗАД">
   <!--Удаление переменных из контекста-->
   <clear keys="#selected,name" />
   <goto target="meters_screen" />
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
  <action type="Exit" title="ВЫХОД">
   <goto target="exit" />
  </action>
 </actions>
</screen>
```

4.6.22 ГРУППОВОЙ ЭКРАН С ЧЕК-БОКСОМ (GROUP/CHECKBOX)

Тип экрана **`type=«group/checkbox»`** используется для реализации чек-бокса на ТПО 5 версии. На ТПО 7 чек-бокс реализуется при помощи поля `<checkbox-field>`, подробнее см. в разделе 5.9.

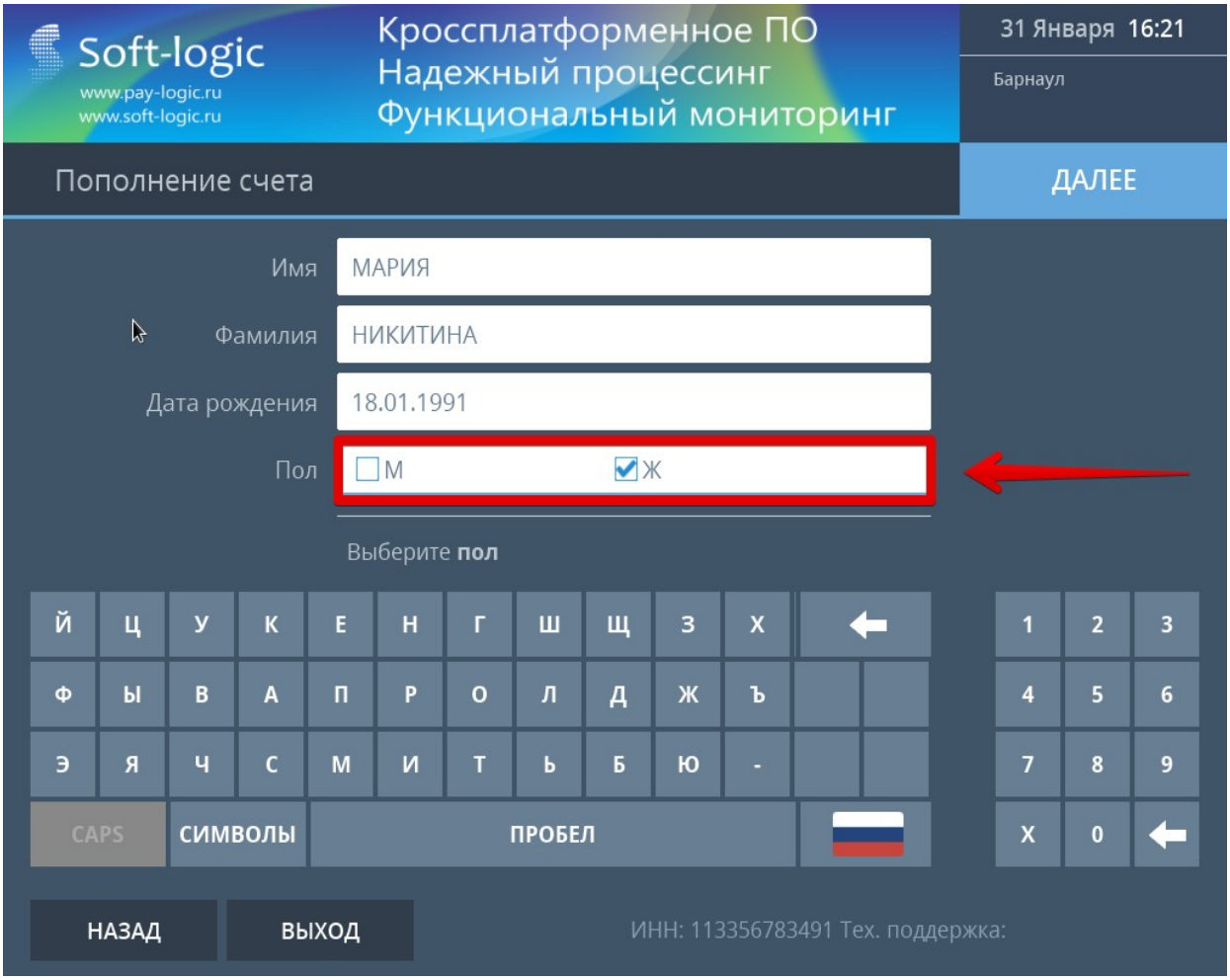


Рисунок 4.6.22.1 — Пример экрана GROUP/CHECKBOX для интерфейса Smoke

Пример:

```
<screen type="group/checkbox" title="Выберите пол" id="reg_client">
       <fields>
          <selector-field id="gender" title="Пол">
                <items type="static">
                    <item title="М" value="men"/>
                    <item title="Ж" value="women"/>
                </items>
                <help>
                    <![CDATA[<html>Выберите <b>пол</b>]]>
                </help>
            </selector-field>
       </fields>
```

Чек-бокс реализуется с помощью `<selector-field>`, который состоит из двух элементов `<item>`. Подробное описание селектора и атрибутов для его элементов см. в разделе 5.10.1.

4.6.23 ИНФОРМАЦИОННЫЙ ЭКРАН (INFO)

Экран с типом **type=«Info»** предназначен для вывода вспомогательной информации (рисунки 4.6.23.1, 4.6.23.2). Для информационного экрана дополнительно доступен элемент `<field>` с атрибутом **key**, который содержит информацию для отображения. Поля ввода на данном типе экрана не используются. Возможно использование html-тегов для форматирования. Кроме того, предусмотрена возможность воспроизвести на экране изображение.

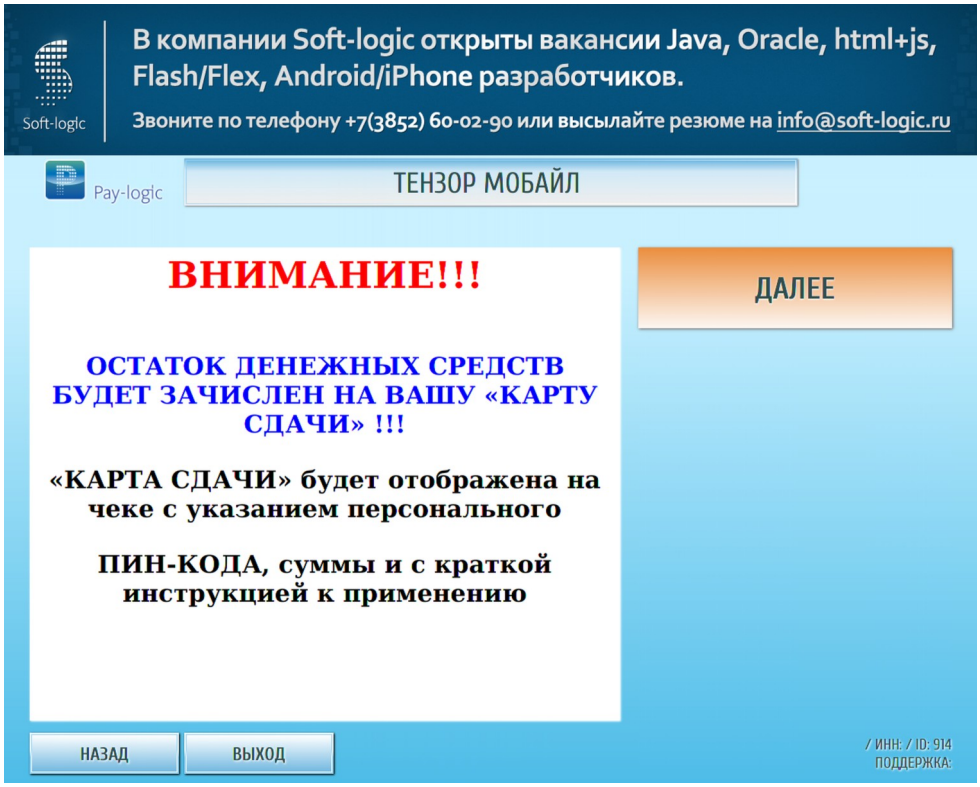


Рисунок 4.6.23.1 — Пример экрана INFO для интерфейса Blues

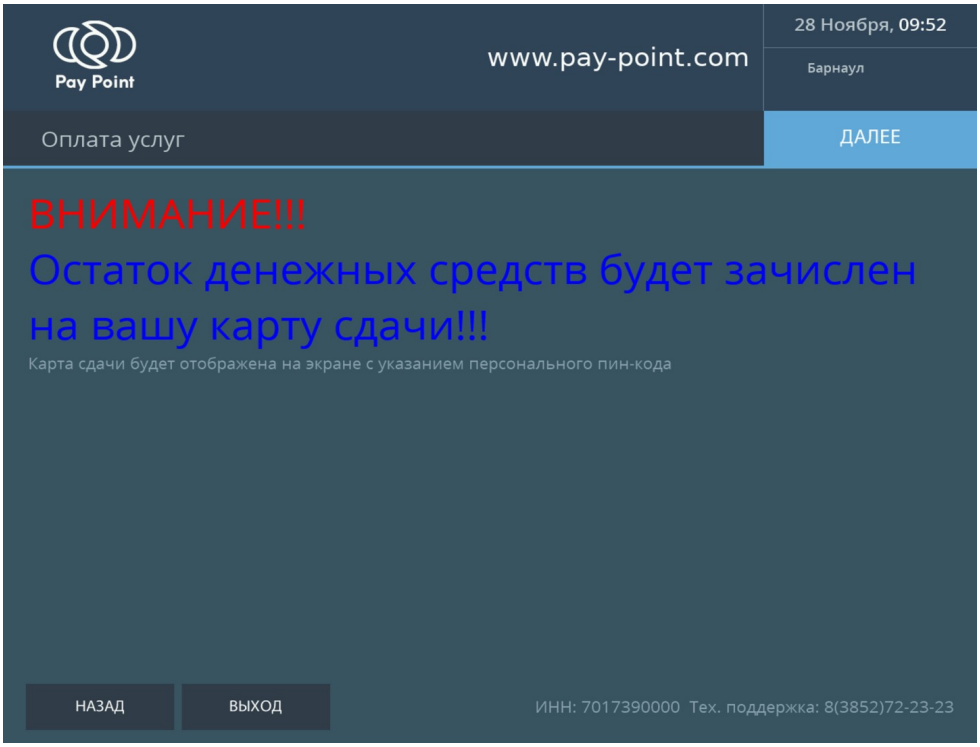


Рисунок 4.6.23.2 — Пример экрана INFO для интерфейса Smoke

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="1screen">
 <!--Создание информационного экрана-->
 <screen type="info" title ="Введите данные" id="1screen">
  <field>
    <![CDATA[<html><font size="10" color="red" weight="bold">ВНИМАНИЕ!!!
     </font> <br><font size="10" color="blue" weight="bold">Остаток
     денежных средств будет зачислен на вашу карту сдачи!!! </font> <br>
     Карта сдачи будет отображена на экране с указанием персонального
     пин-кода]]>
  </field>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <goto-action type="Next" title="Далее" target="pay"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <goto-action type="Prev" title="Назад" target="previous"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
```

```
  <goto-action type="Exit" title="Выход" target="exit"/>
 </actions>
</screen>
...
</scenario>
```

4.6.24 ЭКРАН СОГЛАСИЯ (INFO/AGREE)

Экран с типом **type=«info/agree»** предназначен для вывода плательщику информации об условиях оплаты, с которыми он может согласиться или не согласиться.

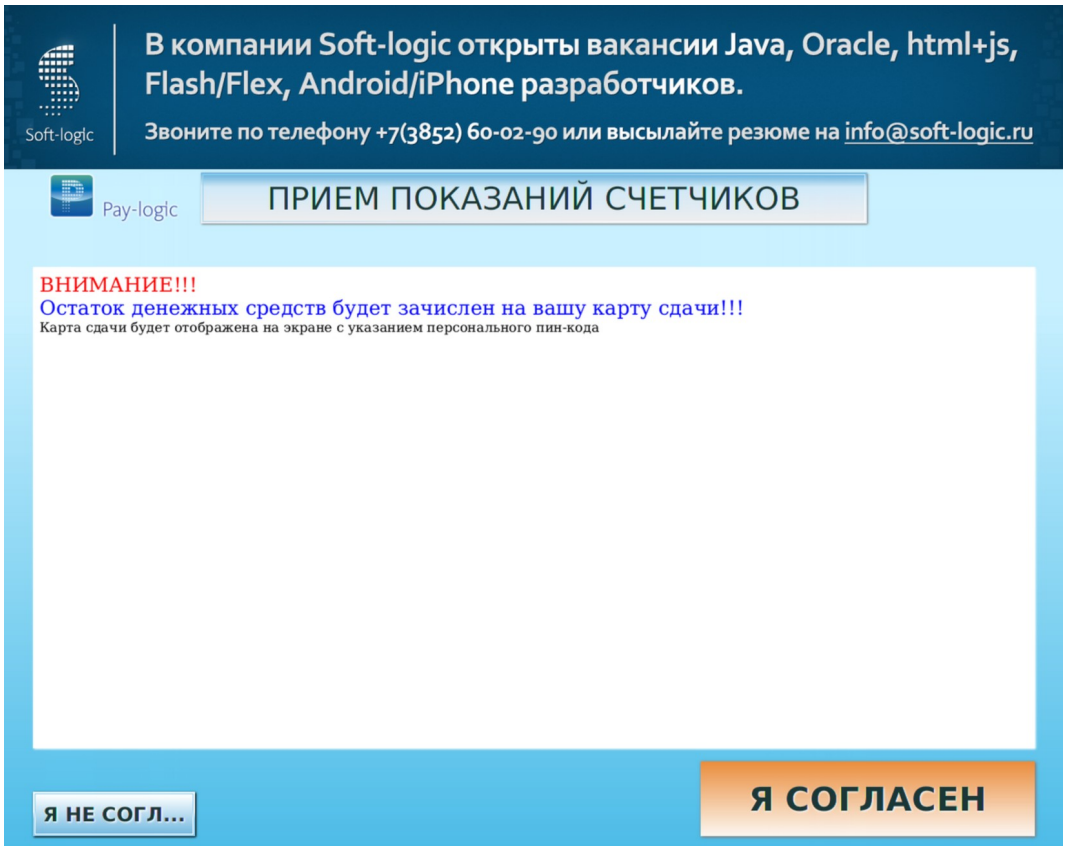


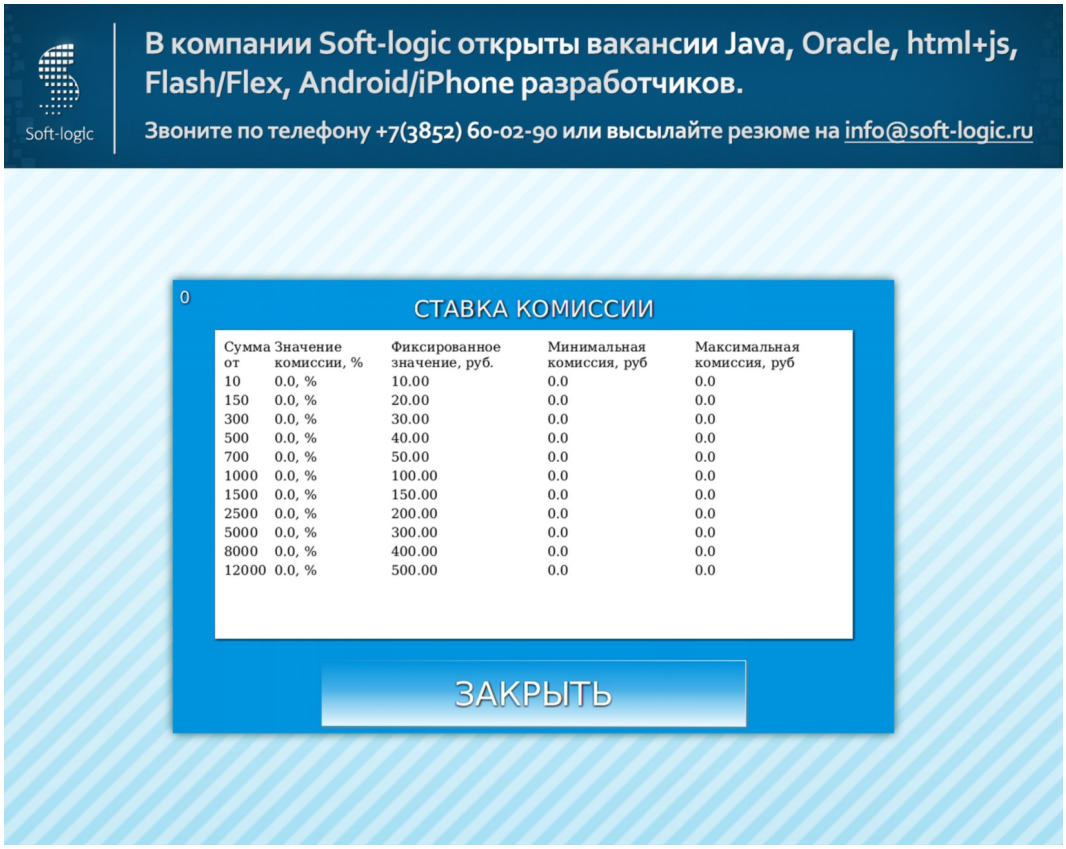
Рисунок 4.6.24.1 — Пример экрана INFO/AGREE для интерфейса Blues

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="1screen">
 <!--Создание информационного экрана-->
 <screen type="info/agree" title ="Введите данные" id="1screen">
  <field>
    <![CDATA[<html> <font size="5" color="red" weight="bold">ВНИМАНИЕ!!!
     </font> br><font size="5" color="blue" weight="bold">Остаток
      денежных средств будет зачислен на вашу карту сдачи!!! </font>
     <br> Карта сдачи будет отображена на экране с указанием
     персонального пин-кода]]>
  </field>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <goto-action type="Next" title="Далее" target="pay"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <goto-action type="Prev" title="Назад" target="previous"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
...
</scenario>
```

4.6.25 ЭКРАН С ИНФОРМАЦИЕЙ О СТАВКЕ КОМИССИИ (INFO/COMM)

Экран с типом **`type=«info/comm»`** предназначен для вывода плательщику информации о действующей комиссии.



Сумма от	Значение комиссии, %	Фиксированное значение, руб.	Минимальная комиссия, руб	Максимальная комиссия, руб
10	0.0, %	10.00	0.0	0.0
150	0.0, %	20.00	0.0	0.0
300	0.0, %	30.00	0.0	0.0
500	0.0, %	40.00	0.0	0.0
700	0.0, %	50.00	0.0	0.0
1000	0.0, %	100.00	0.0	0.0
1500	0.0, %	150.00	0.0	0.0
2500	0.0, %	200.00	0.0	0.0
5000	0.0, %	300.00	0.0	0.0
8000	0.0, %	400.00	0.0	0.0
12000	0.0, %	500.00	0.0	0.0

Рисунок 4.6.25.1 — Пример экрана INFO/COMM для интерфейса Blues

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="1screen">
 <!--Создание информационного экрана с отображением информации о
комиссии-->
 <screen type="info/comm" title ="Введите данные" id="1screen">
  <field>
   <![CDATA[
    <!--Создание таблицы на экране-->
    <table>
     <tr> <td>Сумма от</td><td>Значение комиссии, %</td>
          <td>Фиксированное значение, руб.</td>
          <td>Минимальная комиссия, руб</td>
          <td>Максимальная комиссия, руб</td></tr>
     <tr><td>10</td><td>0.0, %</td><td>10.00</td>
         <td>0.0</td><td>0.0</td></tr>
     <tr><td>150</td><td>0.0, %</td><td>20.00</td>
         <td>0.0</td><td>0.0</td></tr>
     <tr><td>300</td><td>0.0, %</td><td>30.00</td>
         <td>0.0</td><td>0.0</td></tr>
     <tr><td>500</td><td>0.0, %</td><td>40.00</td>
         <td>0.0</td><td>0.0</td></tr>
     <tr><td>700</td><td>0.0, %</td><td>50.00</td>
         <td>0.0</td><td>0.0</td></tr>
     <tr><td>1000</td><td>0.0, %</td><td>100.00</td>
         <td>0.0</td><td>0.0</td></tr>
     <tr><td>1500</td><td>0.0, %</td><td>150.00</td>
         <td>0.0</td><td>0.0</td></tr>
     <tr><td>2500</td><td>0.0, %</td><td>200.00</td>
         <td>0.0</td><td>0.0</td></tr>
     <tr><td>5000</td><td>0.0, %</td><td>300.00</td>
         <td>0.0</td><td>0.0</td></tr>
     <tr><td>8000</td><td>0.0, %</td><td>400.00</td>
         <td>0.0</td><td>0.0</td></tr>
     <tr><td>12000</td><td>0.0, %</td><td>500.00</td>
         <td>0.0</td><td>0.0</td></tr>
    </table>
   ]]>
  </field>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <goto-action type="Next" title="Далее" target="2screen"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
```

```
  <goto-action type="Prev" title="Назад" target="previous"/>
 <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
  <goto-action type="Exit" title="Выход" target="exit"/>
 </actions>
 </screen>
</scenario>
```

4.6.26 ЭКРАНЫ ИДЕНТИФИКАЦИИ КЛИЕНТА (INFO/DOC, INFO/PHOTO, INFO/FACE, INFO/IMAGE)

В ситуации, когда оплата по сервису предполагает предварительную идентификацию клиента, возможно использовать экраны с типами **`info/doc`**, **`info/photo`**, **`info/face`**, **`info/image`**. Экраны реализованы только в ТПО 7 и только для интерфейса **Crypto** — реализация для других интерфейсов осуществляется по запросу.

Экраны предоставляют следующий функционал:

1. **`info/doc`** — используется, если необходимо сфотографировать документ, удостоверяющий личность клиента.
2. **`info/face`** — если требуется сфотографировать лицо клиента.
3. **`info/photo`** — если нужна фотография, где будут одновременно лицо клиента и его документ.
4. **`info/image`** — отображает готовую фотографию с возможностью ее переснять.

Примеры экранов приведены ниже.

Запрос вызова веб-камеры описан в разделе 6.9.5, запрос сопоставления лиц на готовых фотографиях (матчинг) — в разделе 6.9.11.

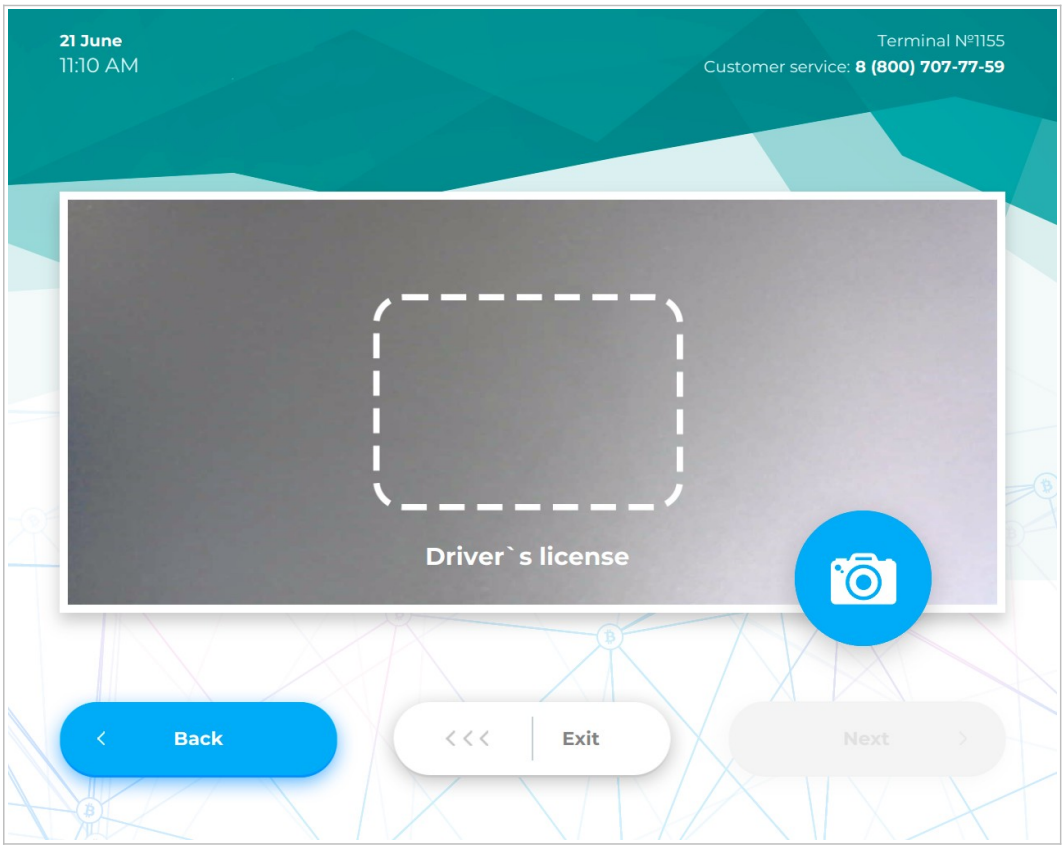


Рисунок 4.6.26.1 — Пример экрана INFO/DOC для интерфейса Crypto

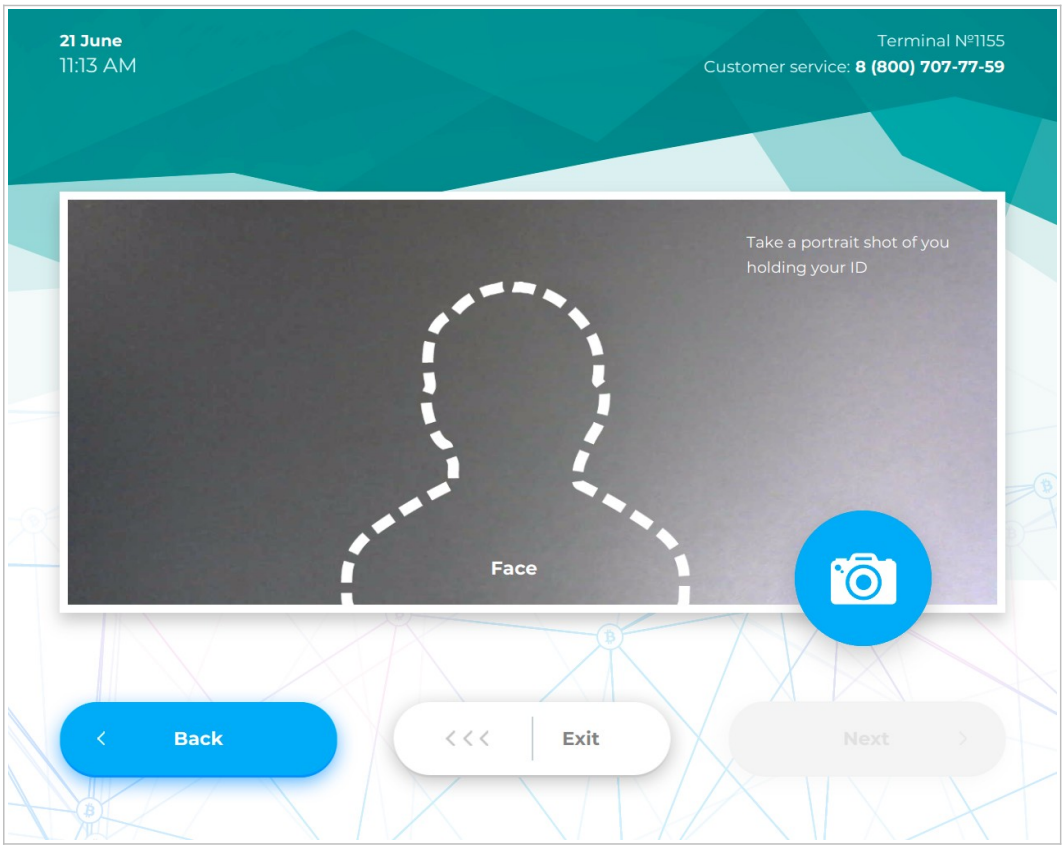


Рисунок 4.6.26.2 — Пример экрана INFO/FACE для интерфейса Crypto

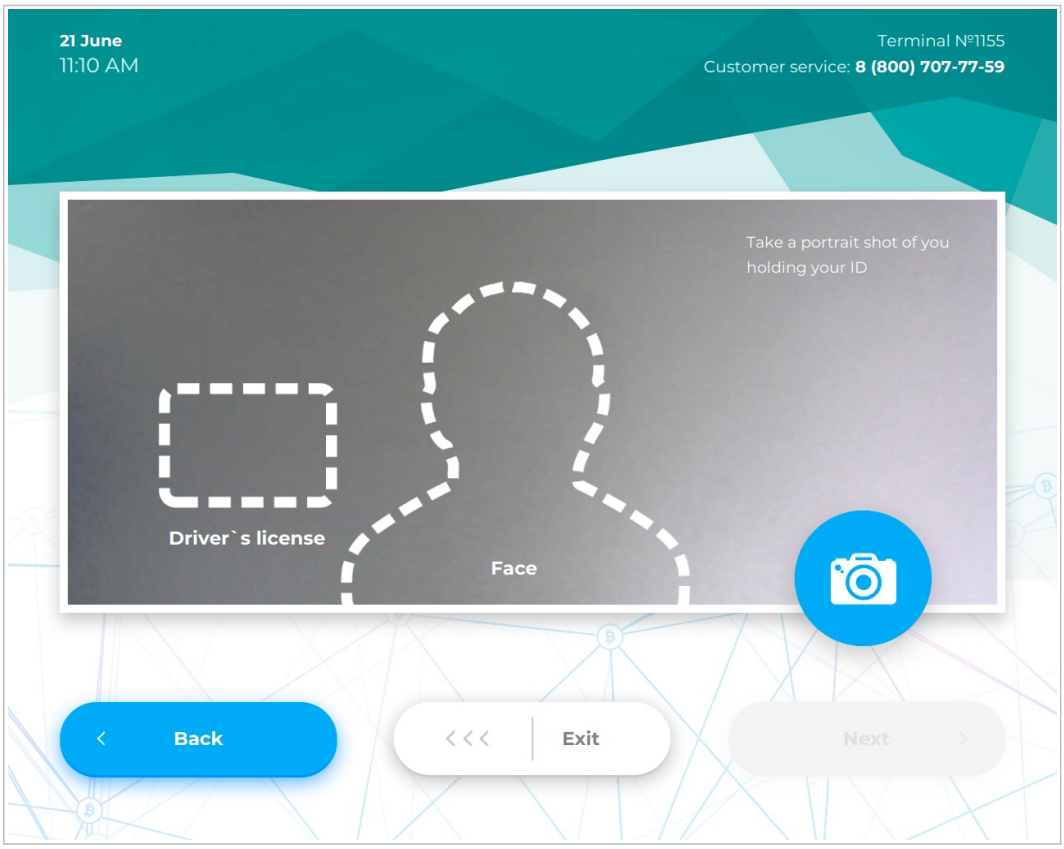


Рисунок 4.6.26.3 — Пример экрана INFO/PHOTO для интерфейса Crypto

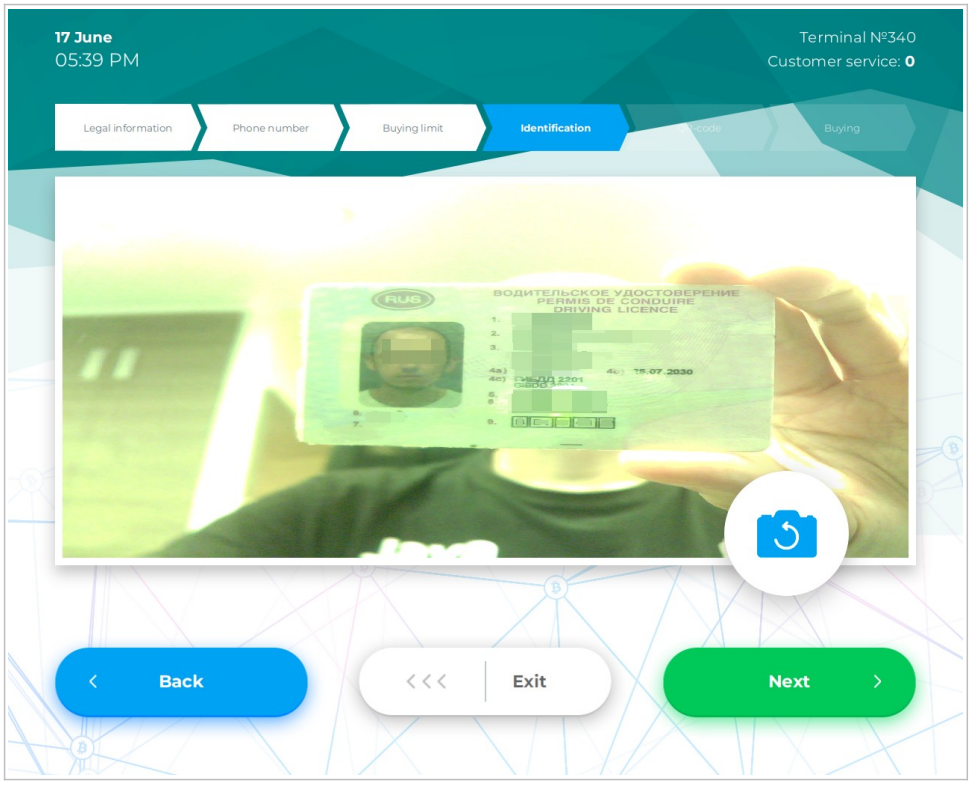


Рисунок 4.6.26.4 — Пример экрана INFO/IMAGE для интерфейса Crypto

Пример:

```
<!-- Создание информационного экрана с возможностью сделать фотографию
документа -->

<screen id="PHOTO_SCAN_ID_DOC" type="info/doc" title="Show your ID">
    <field/>
    <actions>
        <action type="Next" title="Next">

<!-- Запрос к веб-камере терминала -->
            <hdw-request type="webcam" function="stop-video">
                <actions>
                    <action type="success">
                        <goto target="PHOTO_SCAN_ID_DOC_show_photo" />
                    </action>
                    <action type="error">
```

```
                        <dialog type="Info" title="Error"
message="Operation is impossible. Please try again later" timeout="30"
default="okay">
                            <actions>
                                <action type="okay" title="OK">
                                    <cancel />
                                </action>
                            </actions>
                        </dialog>
                    </action>
                </actions>
            </hdw-request>
        </action>
        <action type="screenaction" title="Default">
            <set key="#navi-current" value="identification"/>
            <hdw-request type="webcam" function="start-video">
                <actions>
                    <action type="error">
                        <dialog type="Info" title="Error"
message="Operation is impossible. Please try again later" timeout="30"
default="okay">
                            <actions>
                                <action type="okay" title="OK">
                                    <cancel/>
                                </action>
                            </actions>
                        </dialog>
                    </action>
                </actions>
            </hdw-request>
        </action>
        <action type="Prev" title="Previous">
            <hdw-request type="webcam" function="stop-video">
                <actions>
                    <action type="success">
                        <set key="#back" key-title="The button PREVIOUS
has been pressed" value="thue" value-title="thue" />
                        <goto target="check-identification-screen"/>
                    </action>
                </actions>
            </hdw-request>
        </action>
        <action type="Exit" title="Exit">
```

```
            <hdw-request type="webcam" function="stop-video">
                <actions>
                    <action type="success">
                        <goto target="exit" />
                    </action>
                </actions>
            </hdw-request>
        </action>
    </actions>
</screen>
```

4.6.27 ИНФОРМАЦИОННЫЙ ЭКРАН С НАВИГАЦИЕЙ (INFO/NAVI)

Экран с типом **type=«info/navi»** предназначен для вывода плательщику информации с элементами навигации.

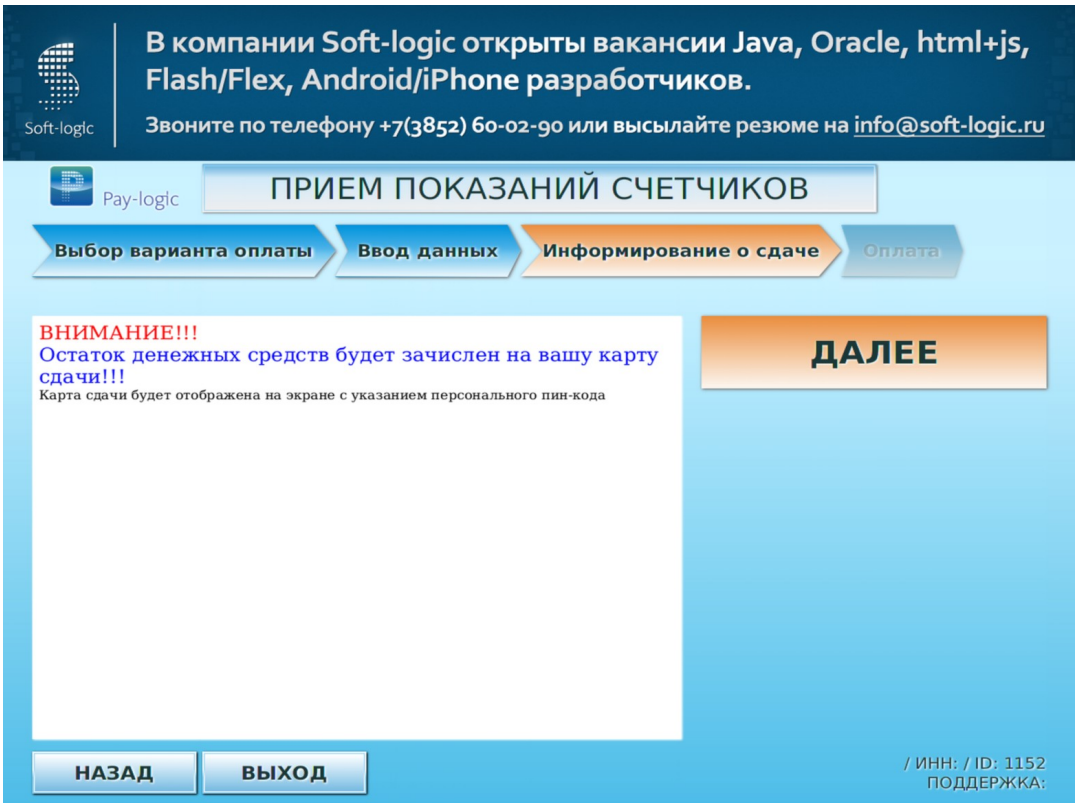


Рисунок 4.6.27.1 — Пример экрана INFO/NAVI для интерфейса Blues

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="1screen">
 <!--Создание информационного экрана-->
 <screen type="info/navi" title ="Введите данные" id="1screen">
  <field>
   <![CDATA[<font size="5" color="red" weight="bold">ВНИМАНИЕ!!!</font>
            <br><font size="5" color="blue" weight="bold">Остаток
```

```
            денежных средств будет зачислен на вашу карту сдачи!!!
            </font> <br>Карта сдачи будет отображена на экране с
            указанием персонального пин-кода]]>
  </field>
  <!--Создание навигации на экране-->
  <navigation>
   <!--Создание первой кнопки навигации-->
   <action type="navi1" title="Выбор варианта оплаты">
    <goto target="0screen"/>
   </action>
   <!--Создание второй кнопки навигации-->
   <action type="navi2" title="Ввод данных">
       <goto target="01screen"/>
   </action>
   <!--Создание третьей кнопки навигации-→
   <action type="navi3" title="Информирование о сдаче" current="true">
    <goto target="1screen"/>
   </action>
   <!--Создание четвертой кнопки навигации-->
   <action type="navi4" title="Оплата">
    <goto target="pay"/>
   </action>
  </navigation>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <goto-action type="Next" title="Далее" target="2screen"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <goto-action type="Prev" title="Назад" target="previous"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
</scenario>
```

4.6.28 ЭКРАН ВВОДА ЧИСЛОВОЙ И ТЕКСТОВОЙ ИНФОРМАЦИИ (NUMERIC)

Экран с типом **type=«numeric»** предназначен для ввода числовой и текстовой информации, рисунки 4.6.28.1, 4.6.28.2. Доступен в ТПО 5 (7im), 7 версии.

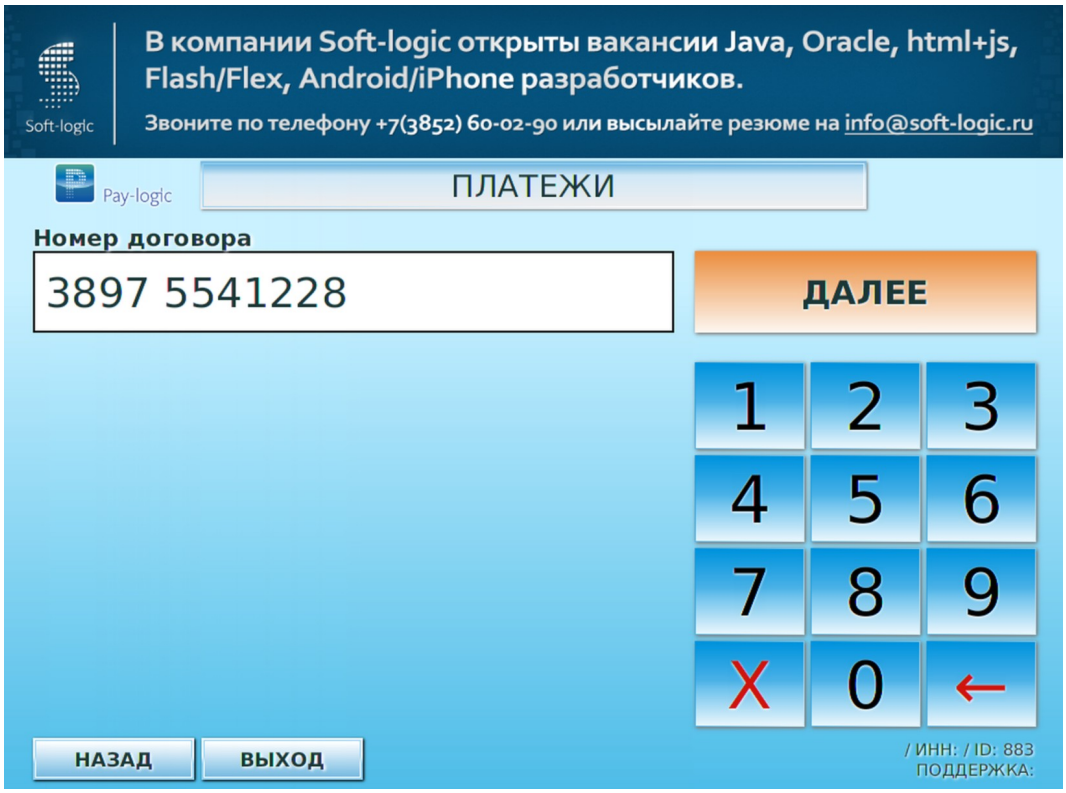


Рисунок 4.6.28.1 — Пример экрана NUMERIC для интерфейса Blues

Для данного типа экрана доступны следующие поля ввода `<text-field>` (раздел 5.7), `<numeric-field>` (раздел 5.8).

В 5 (7im) версии ТПО в качестве заголовка экрана отображается название сервиса, в 7 версии — значение атрибута **title** экрана.

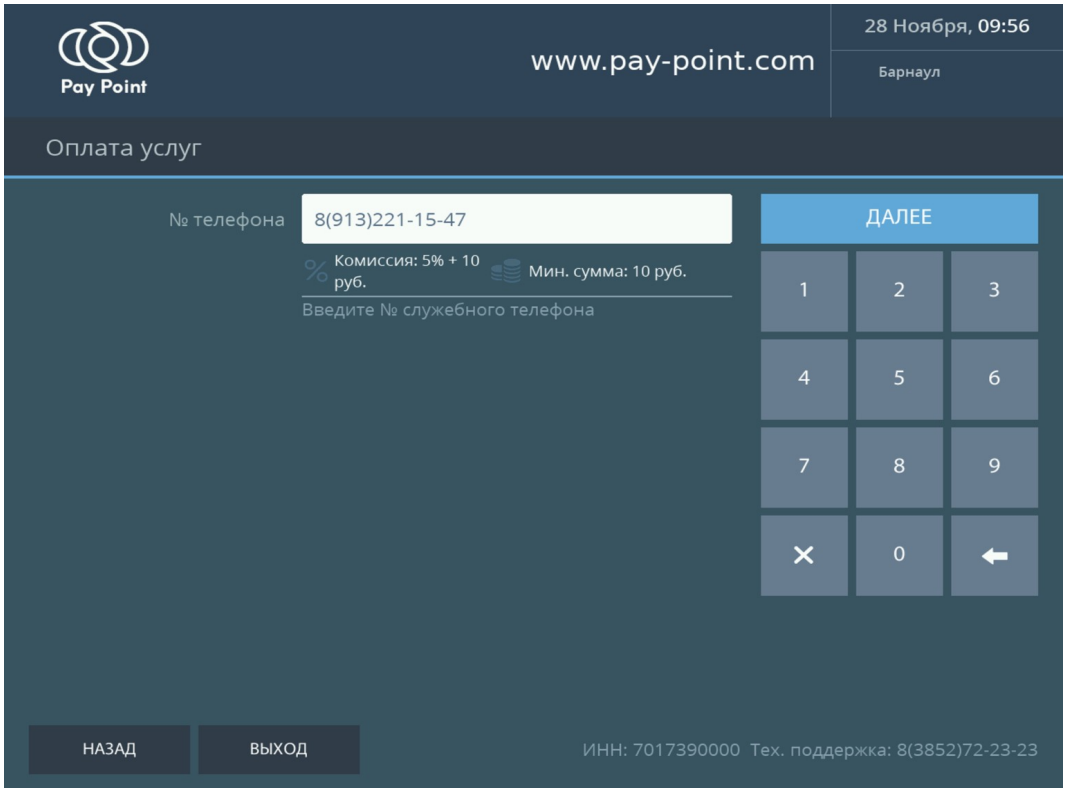


Рисунок 4.6.28.2 — Пример экрана NUMERIC для интерфейса Smoke

4.6.29 ЭКРАН ВВОДА ПИН-КОДА (NUMERIC/CBC)

Экран с типом **`type=«numeric/cbc»`** предназначен для ввода пин-кода карты сдачи (рисунок 4.6.29.1).

Рисунок 4.6.29.1 — Пример экрана NUMERIC/CBC для интерфейса Blues

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="01screen">
 <!--Создание экрана ввода пин-кода-->
 <screen type="numeric/cbc" title="Пин-код карты сдачи" id="01screen">
 <fields>
  <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 16. Название поля
ввода: "Пин-код карты сдачи"-->
  <text-field id="id1" title="Пин-код карты сдачи" max-len="16"
              keyboard="Digital">
   <!--Регулярное выражение для валидации вводимого пин-кода-->
   <validator type="regex">
    <rules> <rule regex="^\d{16}$"/> </rules>
   </validator>
   <!--Регулярное выражение для форматирования вводимого пин-кода на
экране-->
   <formatter type="regex">
    <rules default="****-****-****-****"/>
   </formatter>
   <!--Подсказка, отображается на экране-->
   <help> Введите пин-код карты сдачи </help>
  </text-field>
 </fields>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
  <action type="Next" title="Далее"> <goto target="pay"/> </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
  <action type="Prev" title="Назад"> <goto target="previous"/> </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-→
  <action type="Exit" title="Выход"> <goto target="exit"/> </action>
  </actions>
 </screen>
</scenario>
```

4.6.30 ЧИСЛОВОЙ ЭКРАН С НАВИГАЦИЕЙ (NUMERIC/NAVI)

Экран с типом **type=«numeric/navi»** представляет собой экран для ввода числовых данных с навигацией.

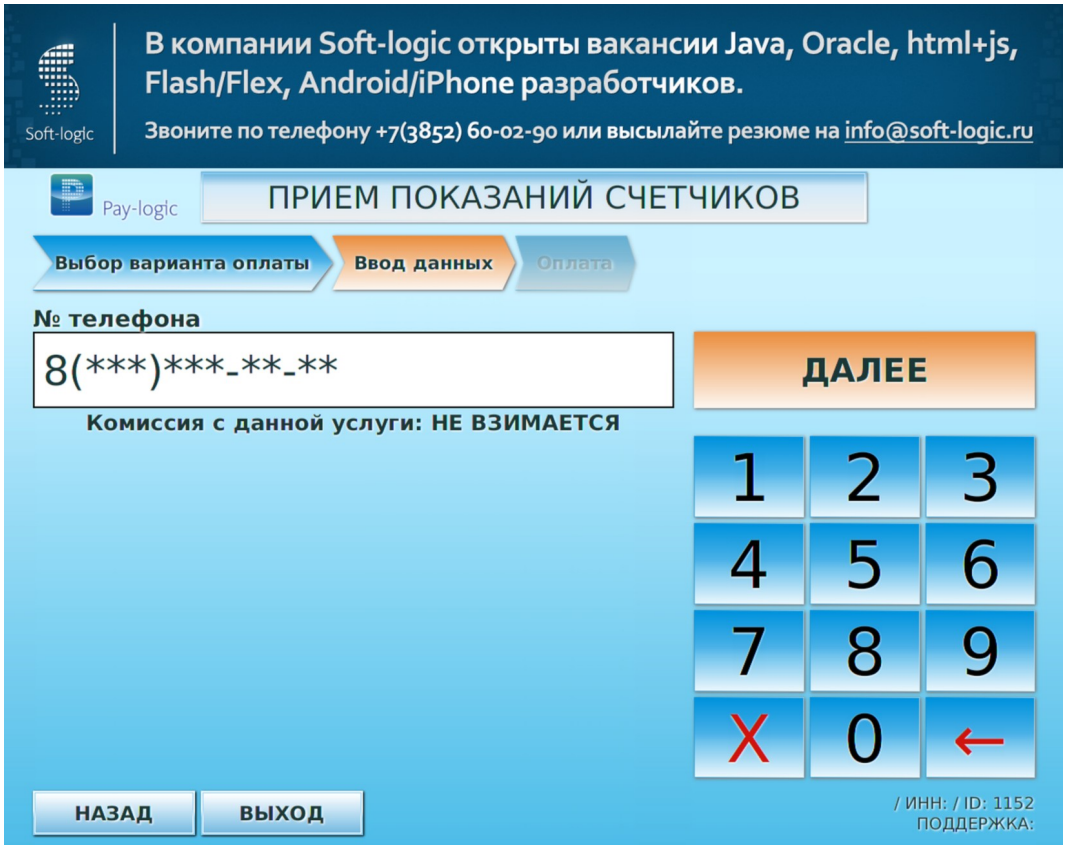


Рисунок 4.6.30.1 — Пример экрана NUMERIC/NAVI для интерфейса Blues

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!--Создание экрана выбора типа оплаты-->
 <screen type="selector" title="Тип оплаты" id="0screen">
  <fields> ... </fields>
  <actions> ... </actions>
 </screen>
```

```
<!-- 01screen -->
<!--Создание экрана ввода числовой и текстовой информации-->
<screen type="numeric" title="Введите № договора" id="01screen">
 <fields> ... </fields>
 <actions> ... </actions>
</screen>
<!-- 02screen-->
<!--Создание экрана ввода данных с несколькими полями-->
<screen type="group" title="Введите № договора,телефона" id="02screen">
 <fields> ... </fields>
 <actions> ... </actions>
</screen>
<!--Создание экрана ввода числовой и текстовой информации с навигацией--
>
<screen type="numeric/navi" title="Введите № телефона" id="03screen">
 <fields>
  <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 10. Название поля
ввода: "№ телефона"-->
  <text-field id="id1" title="№ телефона" max-len="10"
              keyboard="Digital">
   <!--Регулярное выражение для валидации вводимого номера-->
   <validator type="regex">
    <rules> <rule regex="^\d{10}$"/> </rules>
   </validator>
   <!--Регулярное выражение для форматирования вводимого номера на
экране-->
   <formatter type="regex">
    <rules default="8(***)***-**-**"/>
   </formatter>
   <!--Подсказка, отображается на экране-->
   <help> Введите № служебного телефона </help>
  </text-field>
 </fields>
 <navigation>
  <!--Создание третьей кнопки навигации-->
  <action type="navi1" title="Выбор варианта оплаты">
   <goto target="0screen"/>
  </action>
  <!--Создание второй кнопки навигации-->
  <action type="navi2" title="Ввод данных" current="true">
   <goto target="03screen"/>
  </action>
```

```
   <action type="navi3" title="Оплата">
    <goto target="pay"/>
   </action>
  </navigation>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее"> <goto target="pay"/> </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад"> <goto target="0screen"/> </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход"> <goto target="exit"/> </action>
  </actions>
 </screen>
</scenario>
```

4.6.31 ВСПЛЫВАЮЩИЙ ЭКРАН ВВОДА ЧИСЛОВОЙ ИНФОРМАЦИИ (NUMERIC/POPUP)

Экран с типом **`type=«numeric/popup»`** представляет собой экран всплывающий экран ввода числовых данных.

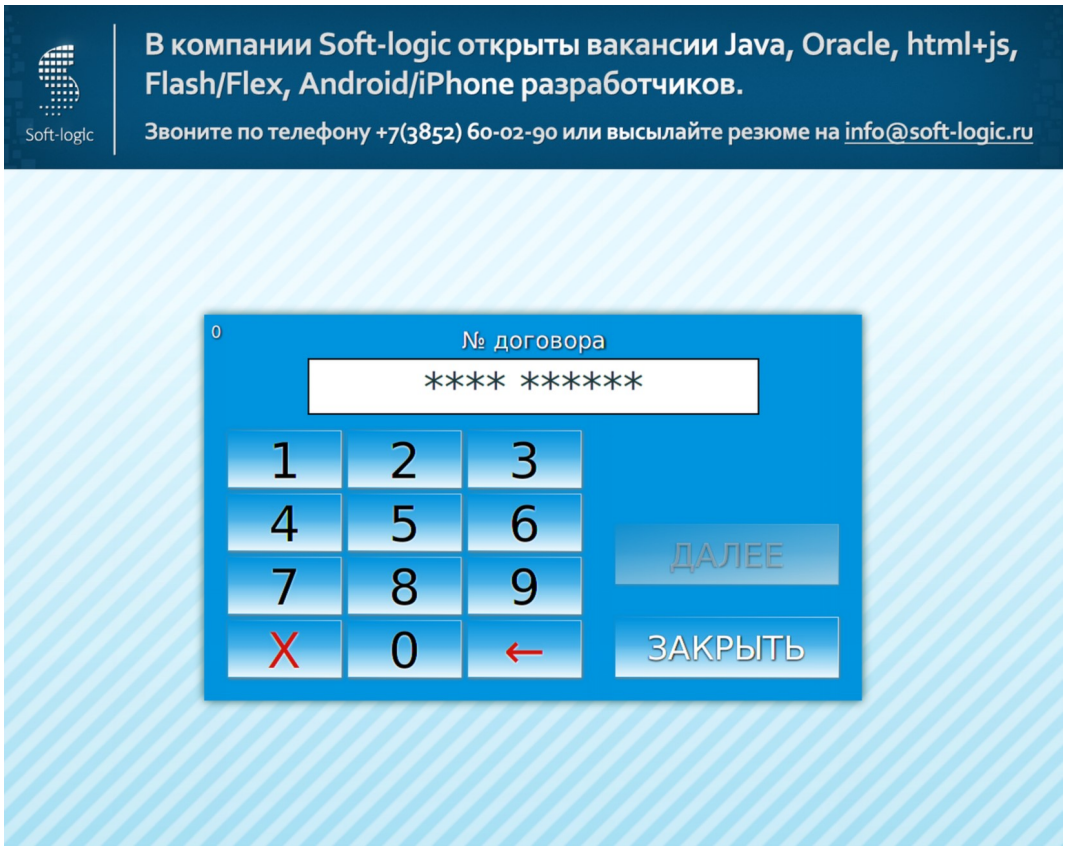


Рисунок 4.6.31.1 — Пример экрана NUMERIC/POPUP для интерфейса Blues

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!--Создание экрана выбора типа оплаты-->
 <screen type="selector" title="Тип оплаты" id="0screen">
  <fields> ... </fields>
  <actions> ... </actions>
 </screen>
 <!-- 01screen -->
 <!--Создание высплывающего экрана ввода числовой и текстовой
информации-->
 <screen type="numeric/popup" title="Введите № договора" id="01screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 11. Название поля
ввода: "№ договора"-->
   <text-field id="id1" title="№ договора" max-len="11"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules> <rule regex="^\d{11}$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого номера на
экране-->
    <formatter type="regex">
     <rules default="**** *****"/>
    </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите № договора </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-→
   <action type="Next" title="Далее">
    <goto target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="0screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход"> <goto target="exit"/> </action>
```

```
  </actions>
 </screen>
 <!-- 02screen →
 <!--Создание экрана ввода с несколькими полями -->
 <screen type="group" title="Введите № договора,телефона" id="02screen">
  <fields>  ... </fields>
  <actions>  ... </actions>
 </screen>
 <!--Создание экрана ввода числовой и текстовой информации-->
 <screen type="numeric/navi" title="Введите № телефона" id="03screen">
  <fields>  ... </fields>
  <navigation>  ... </navigation>
  <actions>  ... </actions>
 </screen>
</scenario>
```

4.6.32 ЭКРАН ВЫБОРА (СЕЛЕКТОР, SELECTOR/LIST)

Экран с типом **`type=«selector/list»`** представляет собой экран выбора значений из списка. Пример экрана приведен на рисунках 4.6.32.1, 4.6.32.2.

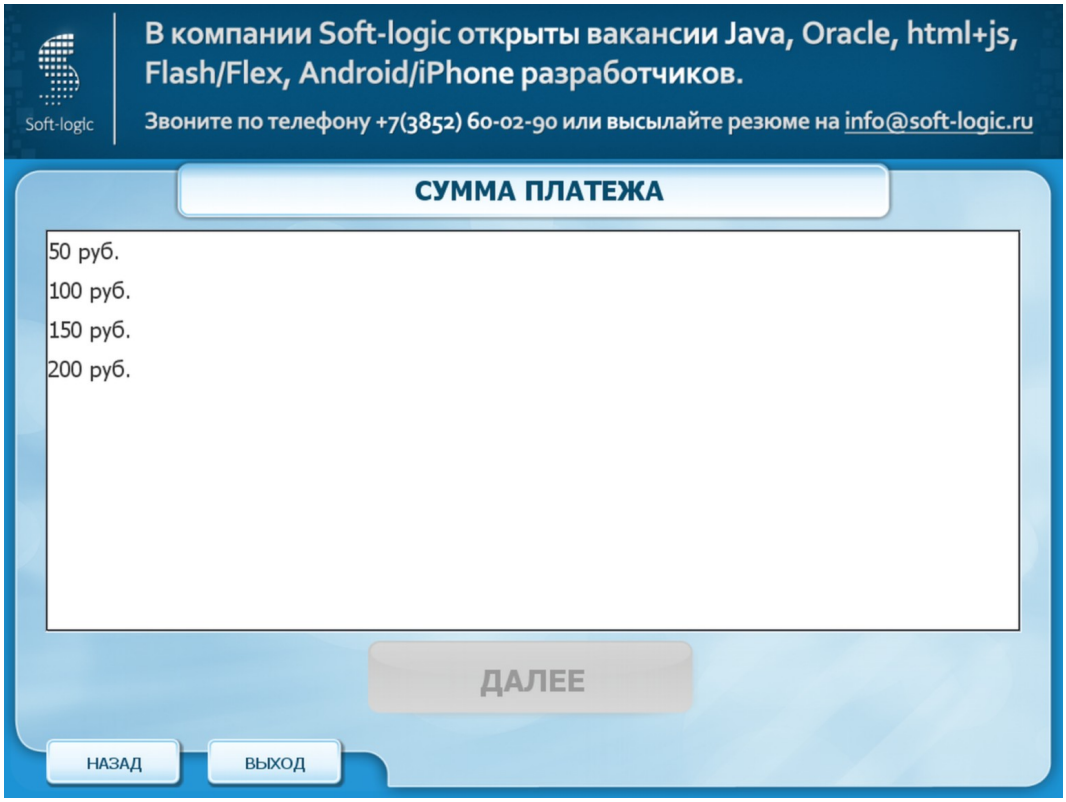


Рисунок 4.6.32.1 — Пример экрана SELECTOR/LIST для интерфейса Blues

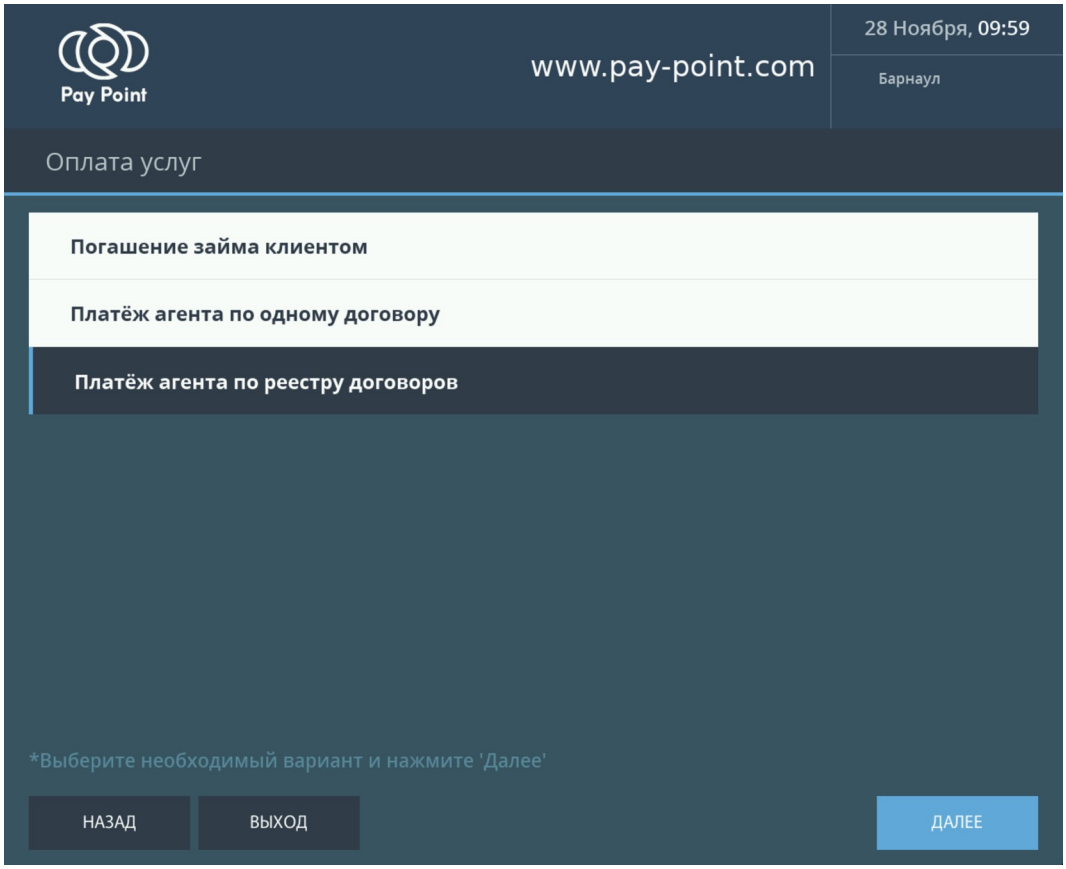


Рисунок 4.6.32.2 — Пример экрана SELECTOR/LIST для интерфейса Smoke

4.6.33 КНОПОЧНЫЙ ЭКРАН ВЫБОРА (SELECTOR/BUTTON)

Экран с типом **type=«selector/button»** представляет собой экран выбора, на котором варианты отображены в виде кнопок. На кнопках могут отображаться изображения.

Пример экрана **type=«selector/button»** приведен на рисунках 4.6.33.1, 4.6.33.2. В 5 версии ТПО в качестве заголовка экрана **«selector»** с **decor=«button»** отображается название сервиса, в 7 версии — значение атрибута **title** экрана.

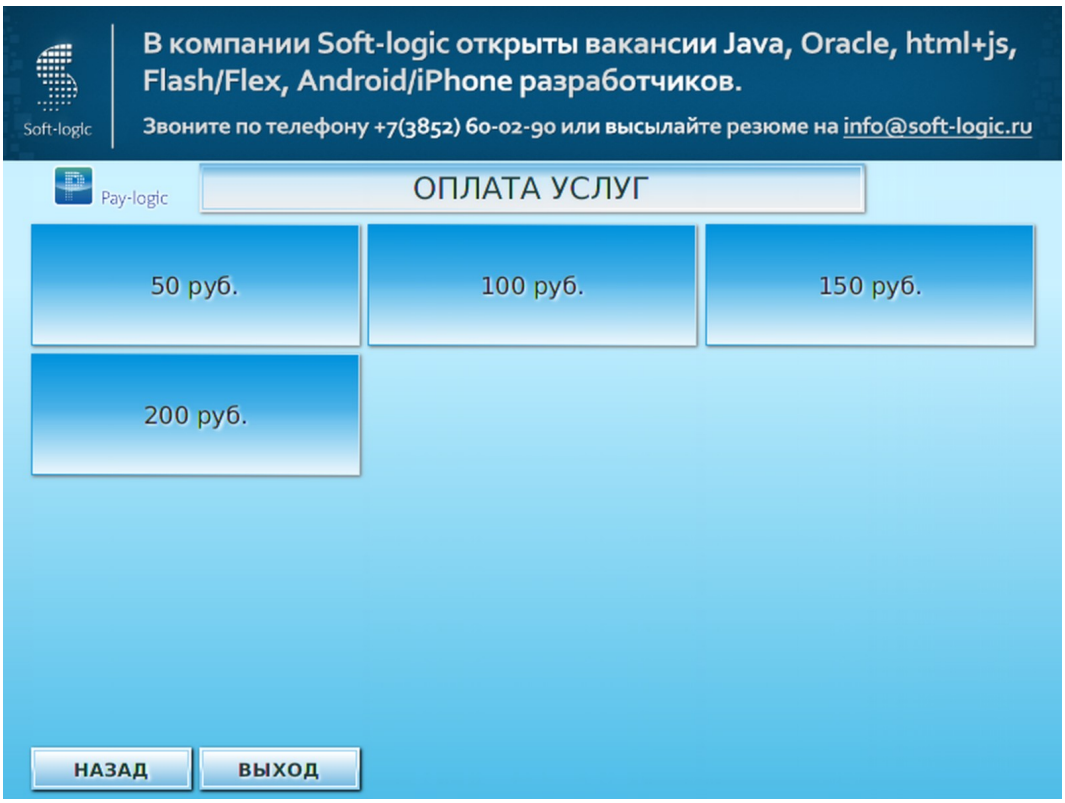


Рисунок 4.6.33.1 — Пример экрана SELECTOR/BUTTON для интерфейса Blues

Рисунок 4.6.33.2 — Пример экрана SELECTOR/BUTTON для интерфейса Smoke

4.6.34 ЭКРАН ВЫБОРА С БОЛЬШИМИ КНОПКАМИ (SELECTOR/BIGBUTTON)

Экран с типом **type=«selector/bigbutton»** аналогичен экрану с типом **type=«selector/button»**, но кнопки на экране больше, чем на экране **type=«selector/button»**.

Рисунок 4.6.34.1 — Пример экрана SELECTOR/BIGBUTTON для интерфейса Smoke

4.6.35 ГРАФИЧЕСКИЙ СЕЛЕКТОР (SELECTOR/CARDS)

В 7 версии ТПО аналогом экрана «SELECTOR» с **`decor=«image»`** является экран «SELECTOR/CARDS».

Рисунок 4.6.35.1 — Пример экрана SELECTOR/CARDS для интерфейса Blues

Пример:

```
<!--Создание экрана выбора-->
<screen id="2_screen" type="selector/cards" title="Мгновенная лотерея">
 <fields>
  <!--Создание поля выбора количества билетов-->
  <selector-field id="vibkol" title="Выбор количества билетов">
   <items type="static">
    <item value="1" image="1ticket.png" sum="13.00"/>
    <item value="2" image="2ticket.png" sum="26.00"/>
    <item value="3" image="3ticket.png" sum="36.00"/>
   </items>
  </selector-field>
 </fields>
<screen>
```

4.6.36 ГРАФИЧЕСКИЙ СЕЛЕКТОР БЕЗ СУММЫ (SELECTOR/IMAGE)

Экран с типом **type=«selector/image»** представляет собой графический селектор, который используется без указания суммы (она может быть произвольной) и без обязательного указания заголовка (присутствует — отображается, отсутствует — не отображается). В 7 версии ТПО изображения должны размещаться в каталоге ***<корень ТПО>/gui/skin/<наименование интерфейса>/img/***. В 5 версии ТПО — ***<корень ТПО>/img/<наименование интерфейса>/icon/service/***. Пример экрана приведен на рисунке 4.6.36.1, 4.6.36.2.

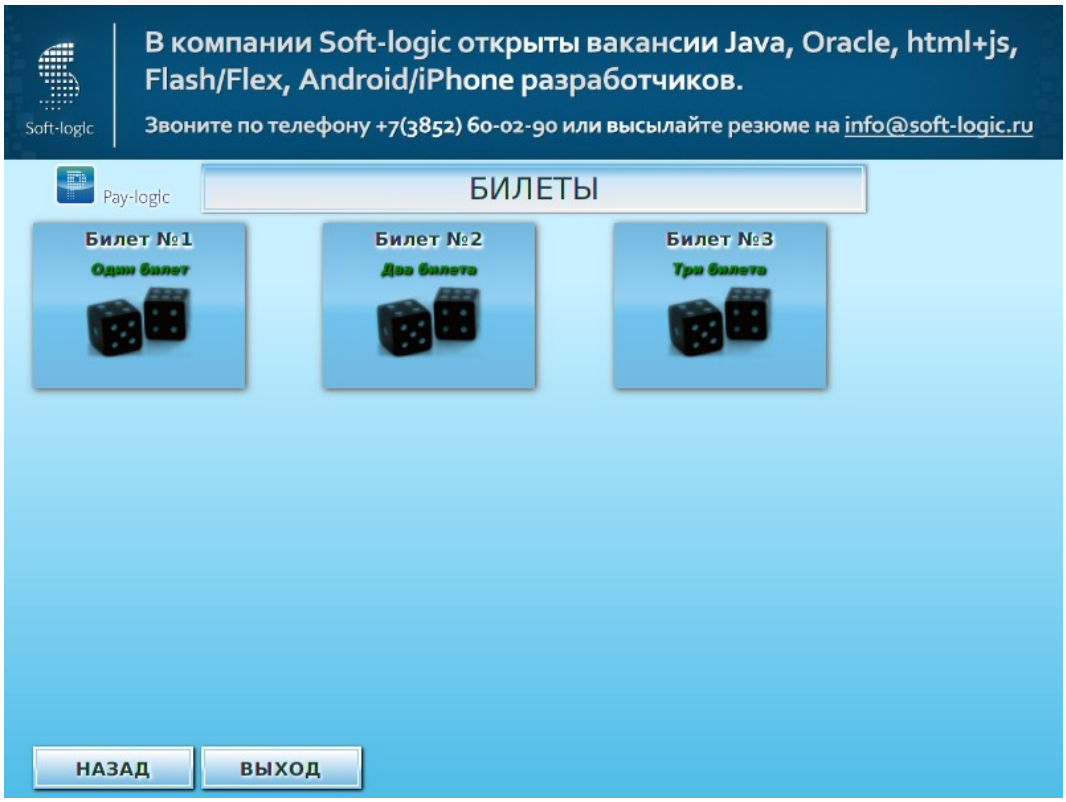


Рисунок 4.6.36.1 — Пример экрана SELECTOR/IMAGE для интерфейса Blues

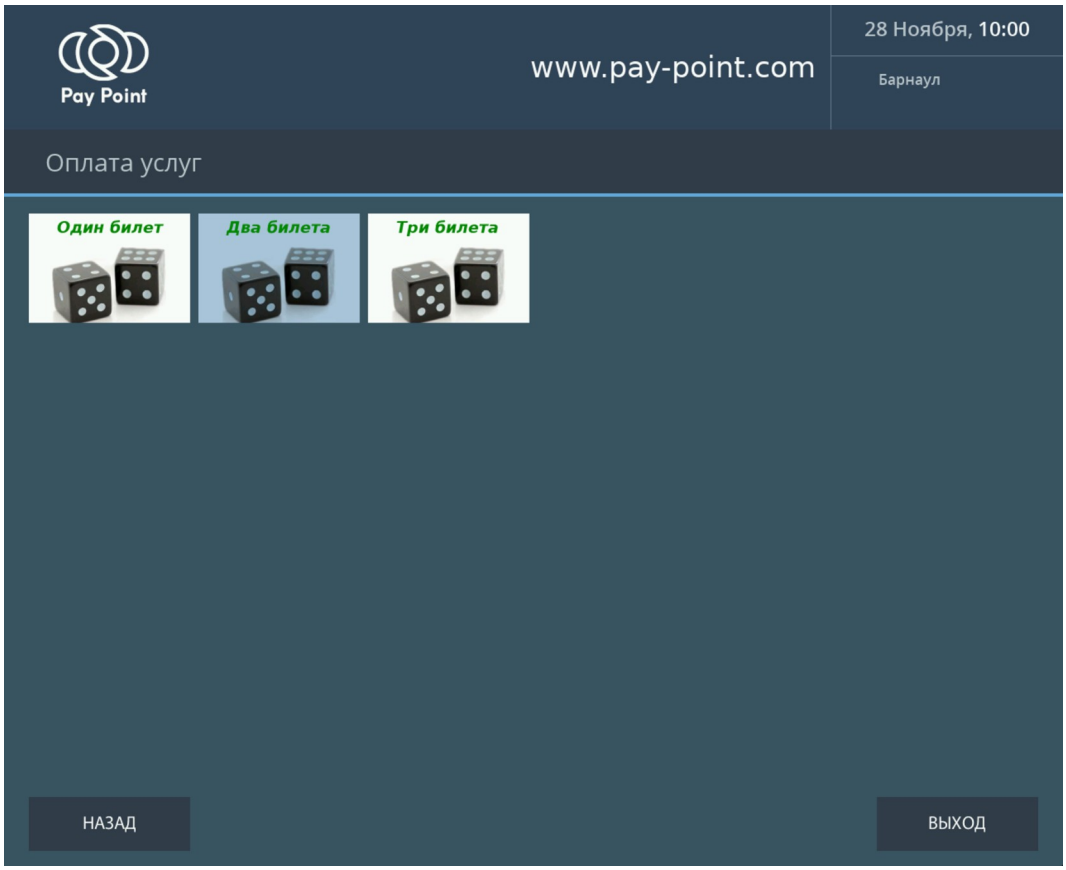


Рисунок 4.6.36.2 — Пример экрана SELECTOR/IMAGE для интерфейса Smoke

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="ids1">
 <!--Создание графического экрана выбора без отображения суммы-->
 <screen type="selector/image" title="selector" id="ids1">
  <fields>
   <!--Создание поля выбора количества билетов-->
   <selector-field id="id1" title="Выбор количества билетов">
    <items type="static">
     <item value="1" title="Билет №1" image="1ticket.png"/>
     <item value="2" title="Билет №2" image="2ticket.png"/>
     <item value="3" title="Билет №3" image="3ticket.png"/>
    </items>
   </selector-field>
  </fields>
```

```
   <actions>
     <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
     <goto-action type="Next" title="ДАЛЕЕ" target="pay"/>
     <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
     <goto-action type="Prev" title="Назад" target="previous"/>
     <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
     <goto-action type="Exit" title="ВЫХОД" target="exit"/>
   </actions>
  </screen>
</scenario>
```

4.6.37 ЭКРАН ВЫБОРА В ВИДЕ СПИСКА (SELECTOR/LIST)

Экран с типом **`type=«selector/list»`** представляет собой экран выбора с вариантами, отображаемыми в виде списка (рисунок 4.6.37.1). В качестве заголовка экрана отображается **`title`** из сценария.

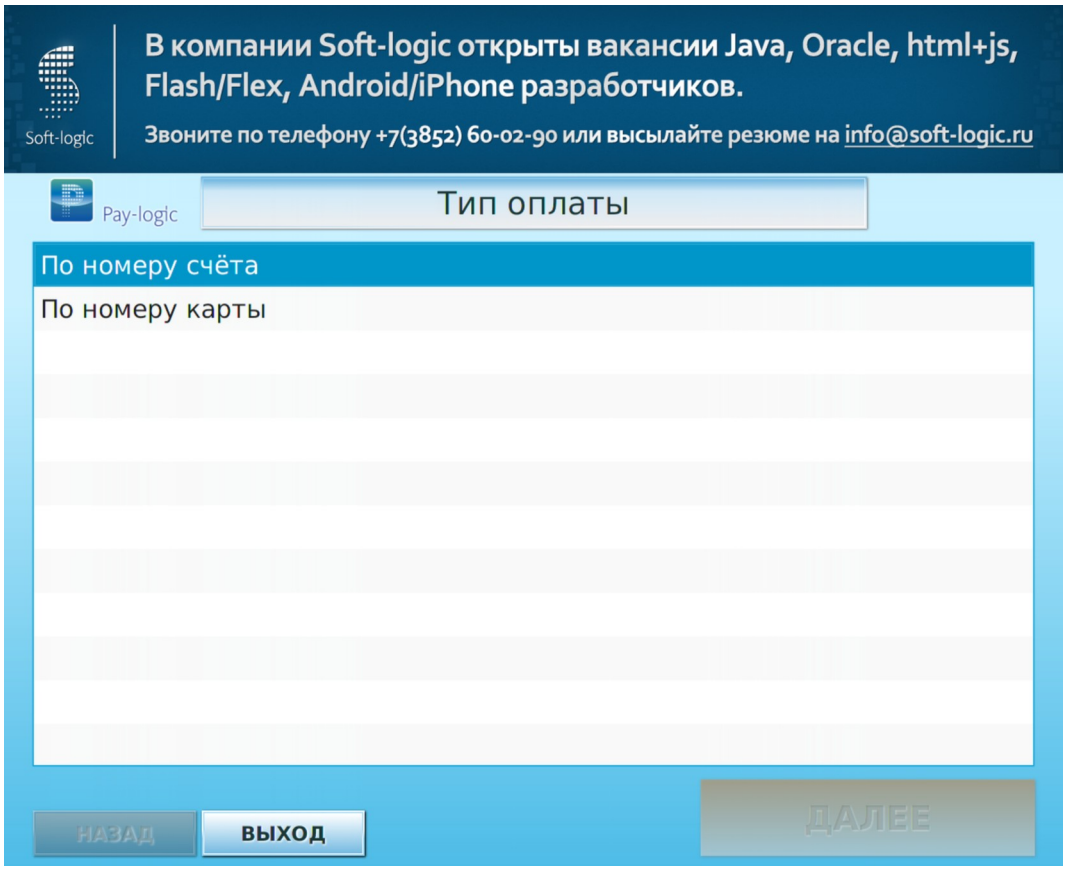


Рисунок 4.6.37.1 — Пример экрана SELECTOR/LIST для интерфейса Blues

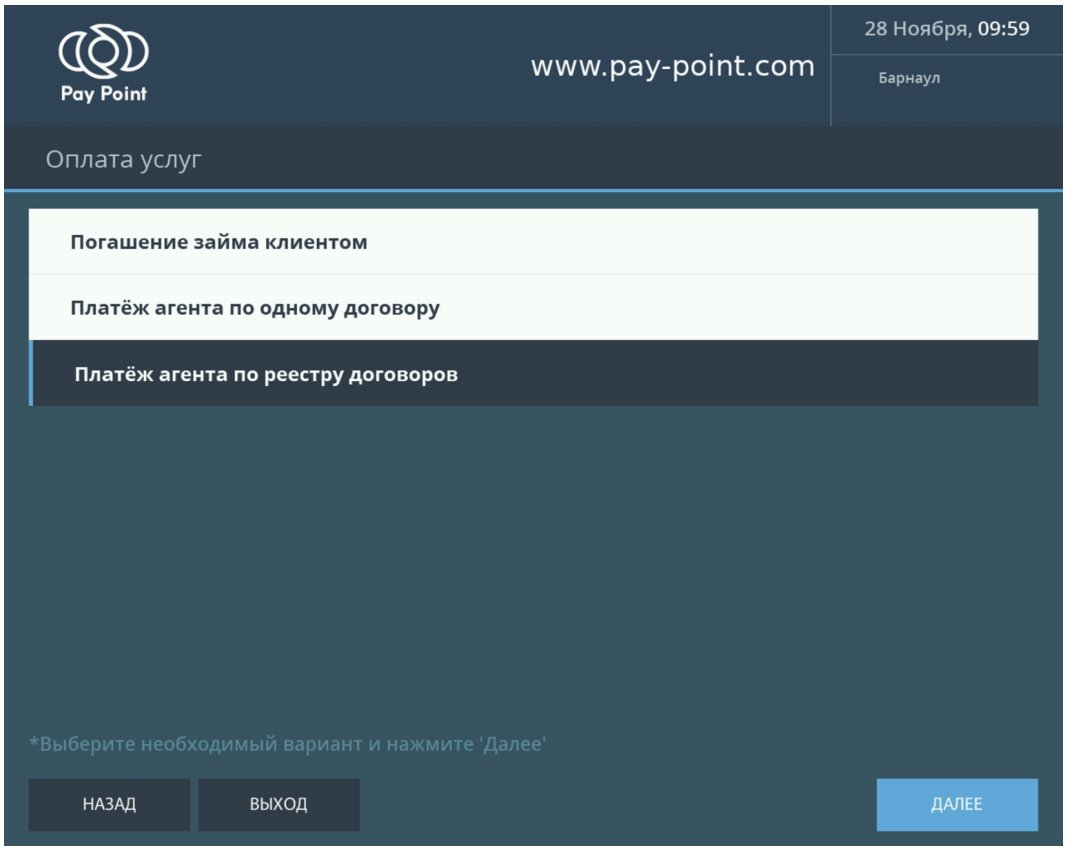


Рисунок 4.6.37.2 — Пример экрана SELECTOR/LIST для интерфейса Smoke

Пример:

```
<!--Создание экрана выбора-->
<screen type="selector/list" title="Тип оплаты" id="0screen">
 <fields>
  <!--Создание поля выбора типа оплаты-->
  <selector-field id="tip" title="Тип оплаты">
   <items type="static">
    <item value="100" title="По номеру счёта"/>
    <item value="200" title="По номеру карты"/>
   </items>
  </selector-field>
 </fields>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
  <action type="Next" title="ДАЛЕЕ">
```

```
   <!--Если значение tip равно 100-->
   <if condition = "tip == 100">
    <!--Переход на экран с id 01screen-->
    <then> <goto target="01screen"/> </then>
    <!--Иначе: переход на экран с id 02screen-->
    <else> <goto target="02screen"/> </else>
   </if>
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
  <action type="Exit" title="ВЫХОД">
   <goto target="exit"/>
  </action>
 </actions>
</screen>
```

4.6.38 ЭКРАН ВЫБОРА В ВИДЕ СПИСКА С НАВИГАЦИЕЙ (SELECTOR/LIST/NAVI)

Экран с типом **`type=«selector/list/navi»`** предназначен для вывода вариантов выбора в виде списка с отображением навигации.

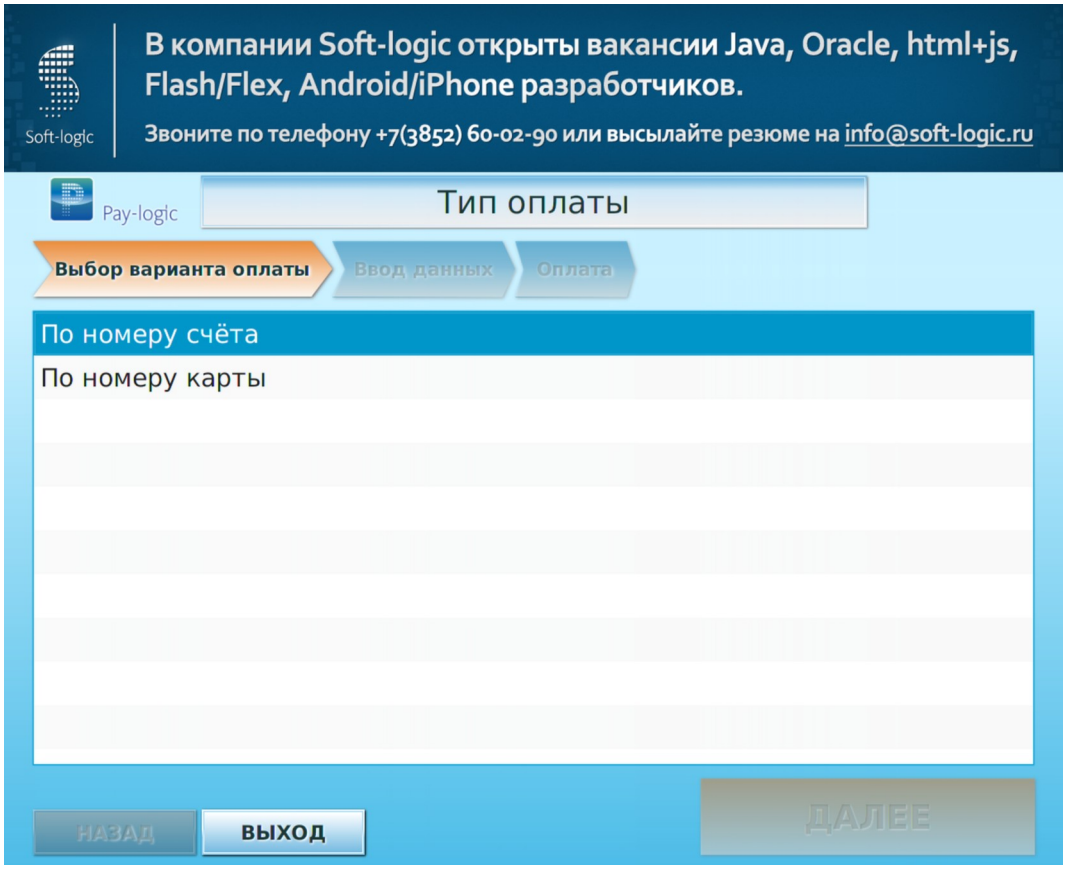


Рисунок 4.6.38.1 — Пример экрана SELECTOR/LIST/NAVI для интерфейса Blues

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!--Создание экрана выбора с навигацией-->
 <screen type="selector/list/navi" title="Тип оплаты" id="0screen">
  <fields>
```

```
  <!--Создание поля выбора типа оплаты-->
  <selector-field id="tip" title="Тип оплаты">
   <items type="static">
    <item value="100" title="По номеру счёта"/>
    <item value="200" title="По номеру карты"/>
   </items>
  </selector-field>
 </fields>
 <!--Создание навигации на экране-->
 <navigation>
  <!--Создание первой кнопки навигации-->
  <action type="navi1" title="Выбор варианта оплаты" current="true">
   <goto target="0screen"/>
  </action>
  <!--Создание второй кнопки навигации-->
  <action type="navi2" title="Ввод данных" >
   <!--Если значение tip равно 100-->
   <if condition = "tip == 100">
    <!--Переход на экран с id 01screen-->
    <then> <goto target="01screen"/> </then>
    <!--Иначе: переход на экран с id 02screen-->
    <else> <goto target="02screen"/> </else>
   </if>
  </action>
  <!--Создание третьей кнопки навигации-->
  <action type="navi3" title="Оплата"> <goto target="pay"/> </action>
 </navigation>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
  <action type="Next" title="ДАЛЕЕ">
   <!--Если значение tip равно 100-->
   <if condition = "tip == 100">
    <!--Переход на экран с id 01screen-->
    <then> <goto target="01screen"/> </then>
    <!--Иначе: переход на экран с id 02screen-->
    <else> <goto target="02screen"/> </else>
   </if>
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
  <action type="Exit" title="ВЫХОД">
   <goto target="exit"/>
  </action>
 </actions>
```

```
 </screen>
</scenario>
```

4.6.39 ЭКРАН ВЫБОРА С НАВИГАЦИЕЙ (SELECTOR/NAVI)

Экран с типом **type=«selector/navi»** предназначен для вывода вариантов выбора с отображением навигации.

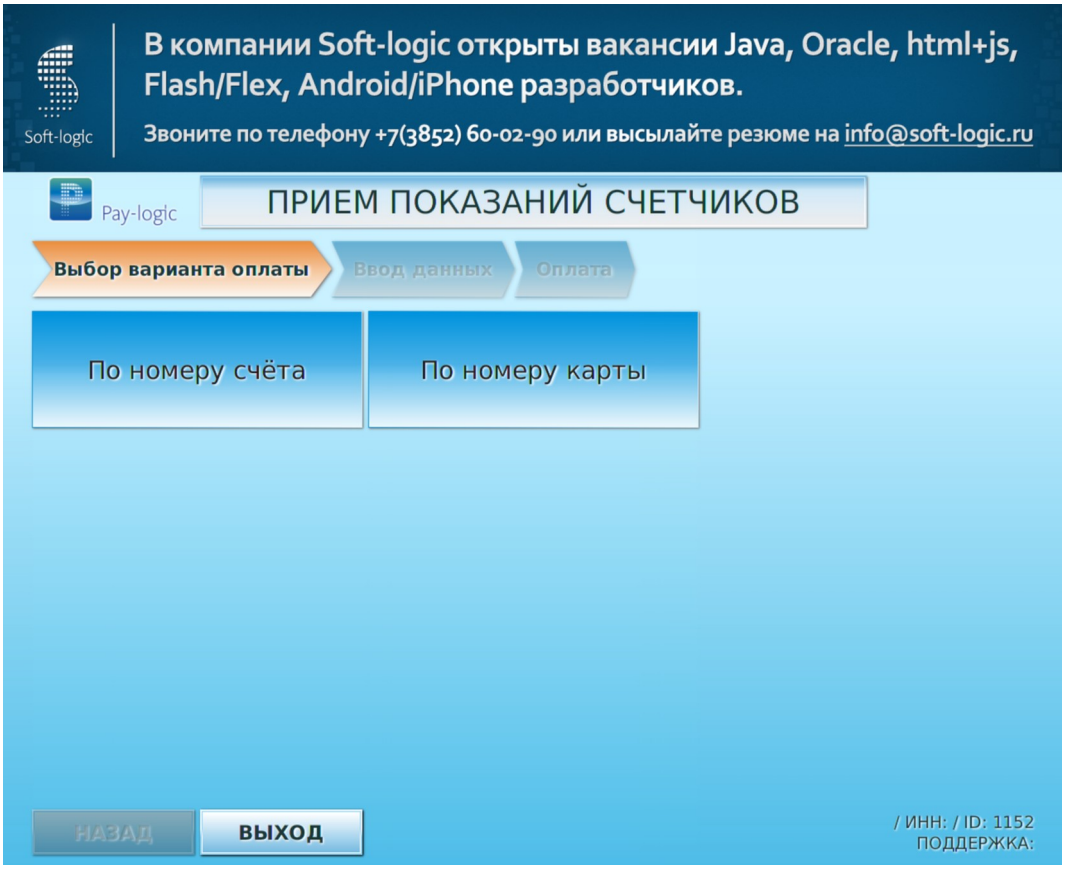


Рисунок 4.6.39.1 — Пример экрана SELECTOR/NAVI для интерфейса Blues

Пример:

```
<!--Создание экрана выбора с навигацией-->
<screen type="selector/navi" title="Тип оплаты" id="0screen">
 <fields>
  <!--Создание поля выбора типа оплаты-->
```

```
 <selector-field id="tip" title="Тип оплаты">
  <items type="static">
   <item value="100" title="По номеру счёта"/>
   <item value="200" title="По номеру карты"/>
  </items>
 </selector-field>
</fields>
<!--Создание навигации на экране-->
<navigation>
 <!--Создание первой кнопки навигации-->
 <action type="navi1" title="Выбор варианта оплаты" current="true">
  <goto target="0screen"/>
 </action>
 <!--Создание второй кнопки навигации-->
 <action type="navi2" title="Ввод данных" >
  <!--Если значение tip равно 100-→
  <if condition = "tip == 100">
   <!--Переход на экран с id 01screen-->
   <then> <goto target="01screen"/> </then>
   <!--Иначе: переход на экран с id 02screen-->
   <else> <goto target="02screen"/> </else>
  </if>
 </action>
 <!--Создание третьей кнопки навигации-->
 <action type="navi3" title="Оплата">
  <goto target="pay"/>
 </action>
</navigation>
<actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
 <action type="Next" title="ДАЛЕЕ">
  <!--Если значение tip равно 100-→
  <if condition = "tip == 100">
   <!--Переход на экран с id 01screen-->
   <then> <goto target="01screen"/> </then>
   <!--Иначе: переход на экран с id 02screen-->
   <else> <goto target="02screen"/> </else>
  </if>
 </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
 <action type="Exit" title="ВЫХОД">
  <goto target="exit"/>
 </action>
```

```
 </actions>
</screen>
```

4.6.40 ЭКРАН ВЫБОРА ВАРИАНТОМ (SELECTOR/VARIANT)

Экран с типом **type=«selector/variant»** предназначен для вывода вариантов выбора в виде кнопок с пояснением. Возможно размещение четырех кнопок и полей информации справа.

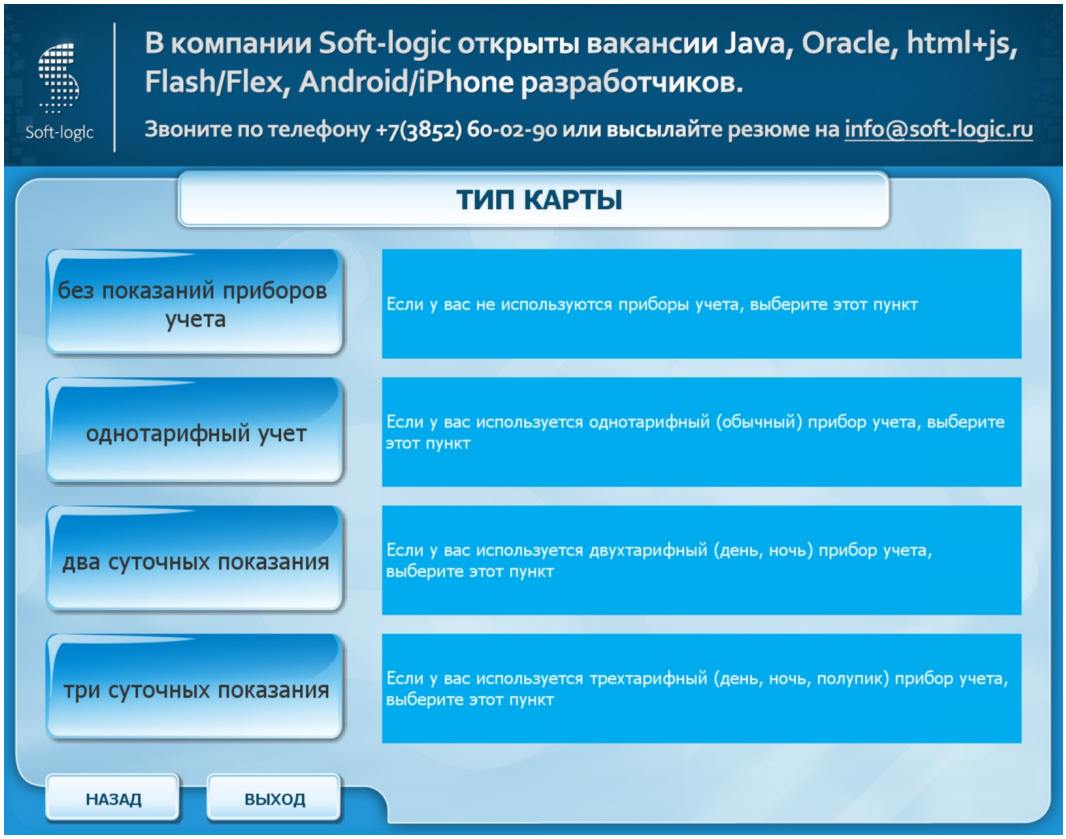


Рисунок 4.6.40.1 — Пример экрана **«selector/variant»** для BlueSphere2

Пример (ТПО 5 (5im)):

```
<!--Создание экрана выбора типа карт-->
<screen type="Selector" decor="variant" title="Тип карты"
        id="first_screen">
 <fields>
  <!--Создание поля выбора-->
  <selector-field id="ev_account1" title="">
   <items type="static">
    <item value="0" title="без показаний приборов учета">
     <!--Подсказка, отображается на экране-->
     <help>
      <![CDATA[
       <html>Если у вас не используются приборы учета, выберите этот
            пункт]]>
     </help>
    </item>
    <item value="1" title="однотарифный учет">
     <!--Подсказка, отображается на экране-->
     <help>
      <![CDATA[
       <html>Если у вас используется однотарифный (обычный) прибор
            учета, выберите этот пункт]]>
     </help>
    </item>
    <item value="2" title="два суточных показания">
     <!--Подсказка, отображается на экране-->
     <help>
      <![CDATA[<html>Если у вас используется двухтарифный (день, ночь)
      прибор учета, выберите этот пункт]]>
     </help>
    </item>
    <item value="3" title="три суточных показания">
     <!--Подсказка, отображается на экране-->
     <help>
      <![CDATA[
       <html>Если у вас используется трехтарифный (день, ночь,
            полупик) прибор учета, выберите этот пункт]]>
     </help>
    </item>
   </items>
  </selector-field>
 </fields>
```

```
 <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
 <action type="Next" title="ДАЛЕЕ">
    <goto target="pay"/>
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
  <action type="Prev" title="НАЗАД">
    <goto target="previous"/>
  </action>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
  <action type="Exit" title="ВЫХОД">
    <goto target="exit"/>
  </action>
 </actions>
</screen>
```

4.6.41 АЛЬТЕРНАТИВНЫЙ ЭКРАН ОПЛАТЫ (SUM)

Экран с типом **type=«sum»** может использоваться для:

1. Переопределения атрибута, отображаемого на экране оплаты. По умолчанию на экране оплаты отображается атрибут **id1**.

2. Переопределения типа экрана оплаты. По умолчанию в интерфейсе используется один стандартный экран оплаты. При необходимости в интерфейс могут быть добавлены другие экраны оплаты, визуально отличающиеся от стандартного. Зная код альтернативного экрана, в сценарии для определенного сервиса возможно переопределить экран оплаты.

3. Одновременного переопределения атрибута, отображаемого на экране оплаты, и используемого экрана оплаты.

Для переопределения атрибута, отображаемого на экране оплаты в атрибуте **title** элемента **<screen>** с **type=«Sum»** укажите атрибут, который требуется отображать пользователю. Атрибут может быть как определен в сценарии, так и возвращен провайдером в результате advanced-запроса.

Пример:

```
<screen type="Sum" title="films"/>
```

Вместо **id1** на экране оплаты будет отображаться значение атрибута **films** — рисунок 4.6.41.1.

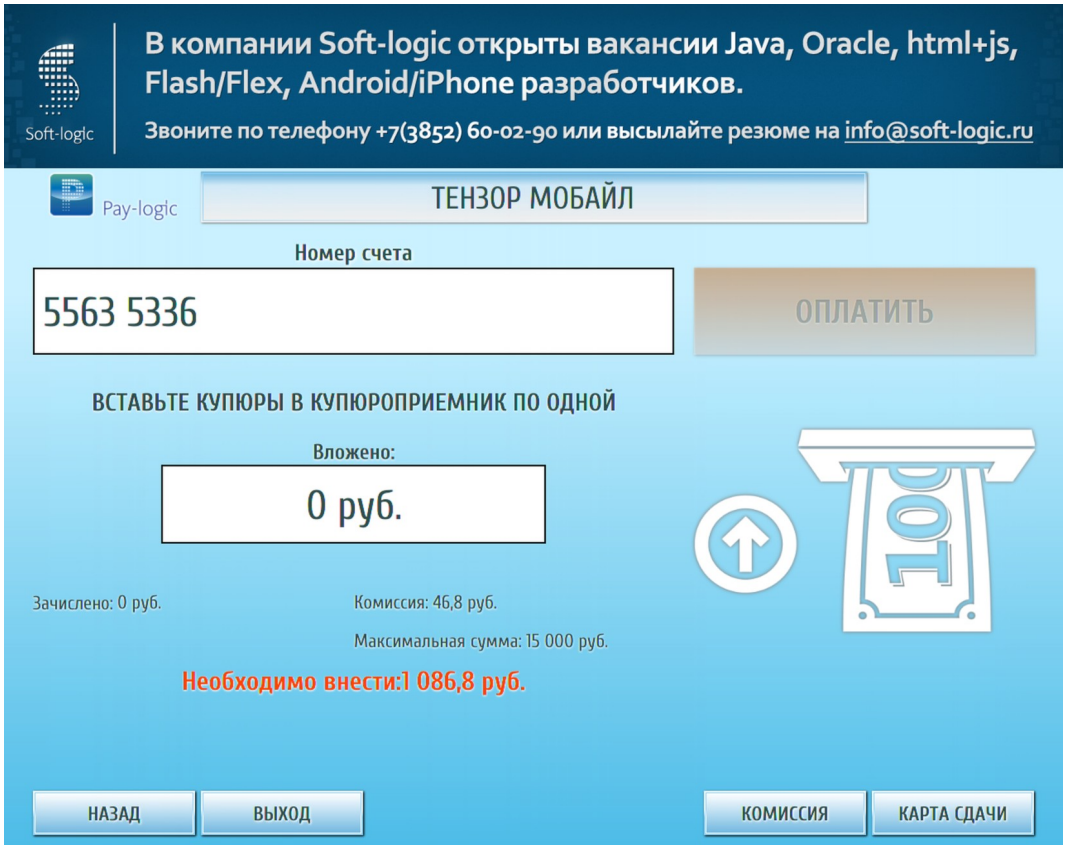


Рисунок 4.6.41.1 — Пример экрана SUM для интерфейса Blues

Для переопределения экрана оплаты в атрибуте **sum** элемента `<screen>` с **type=«Sum»** укажите код экрана, который требуется использовать для данного сервиса. Коды альтернативных экранов для каждого интерфейса уточняйте у сотрудников компании Soft-logic.

Возможно одновременно переопределить атрибут, отображаемый на экране оплаты, и сам экран оплаты. В примере ниже вместо **id1** на экране оплаты отображается

значение атрибута **phone**, а вместо стандартного экрана используется **"cash/communal"**.

Пример (7 версия ТПО):

```
<?xml version="1.0" encoding="UTF-8"?>
<scenario xmlns="http://pay-logic.ru" begin="screen_1">
<!--Переопределение экрана оплата-->
 <screen type="Sum" title="phone" sum="cash/communal"/>
  <!--Создание группового экрана ввода данных-->
  <screen type="group" id="screen_1" title="тел">
   <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 10. Название поля
ввода: "Лицевой счет"-->
   <text-field id="id1" title="Лицевой счет" max-len="10"
               keyboard="Digital" message="Введите лицевой счет">
    <!--Регулярное выражение для валидации лицевого счета-->
    <validator type="regex">
     <rules>
      <rule regex="^\d{1,10}$"/>
     </rules>
    </validator>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 10. Название поля
ввода: "Номер телефона"-->
   <text-field id="phone" title="Номер телефона" max-len="10"
               keyboard="Digital" message="Введите номер телефона">
    <!--Регулярное выражение для валидации введенного номера-→
    <validator type="regex">
     <rules>
      <rule regex="^\d{1,10}$"/>
     </rules>
    </validator>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <goto target="pay"/>
   </action>
```

```
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="previous"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
</scenario>
```

Пример экрана оплаты приведен на рисунке 4.6.41.2[1].

Частным примером переопределения экрана оплаты является использование специального экрана для вывода на нем рекламных баннеров. На стандартном экране оплаты в интерфейсах Bluesphere, Blues отображение рекламных баннеров не предусмотрено.

Пример:

```
<screen type="Sum" sum="sum-banner"/>
```

1 Так как целью данного примера является демонстрация возможностей использования альтернативных экранов оплаты, то в примере не приводится код для заполнения данных на приведенном экране оплаты. Для заполнения данных на таком экране в сценарии должна осуществляться работа с корзиной платежей (раздел 7)

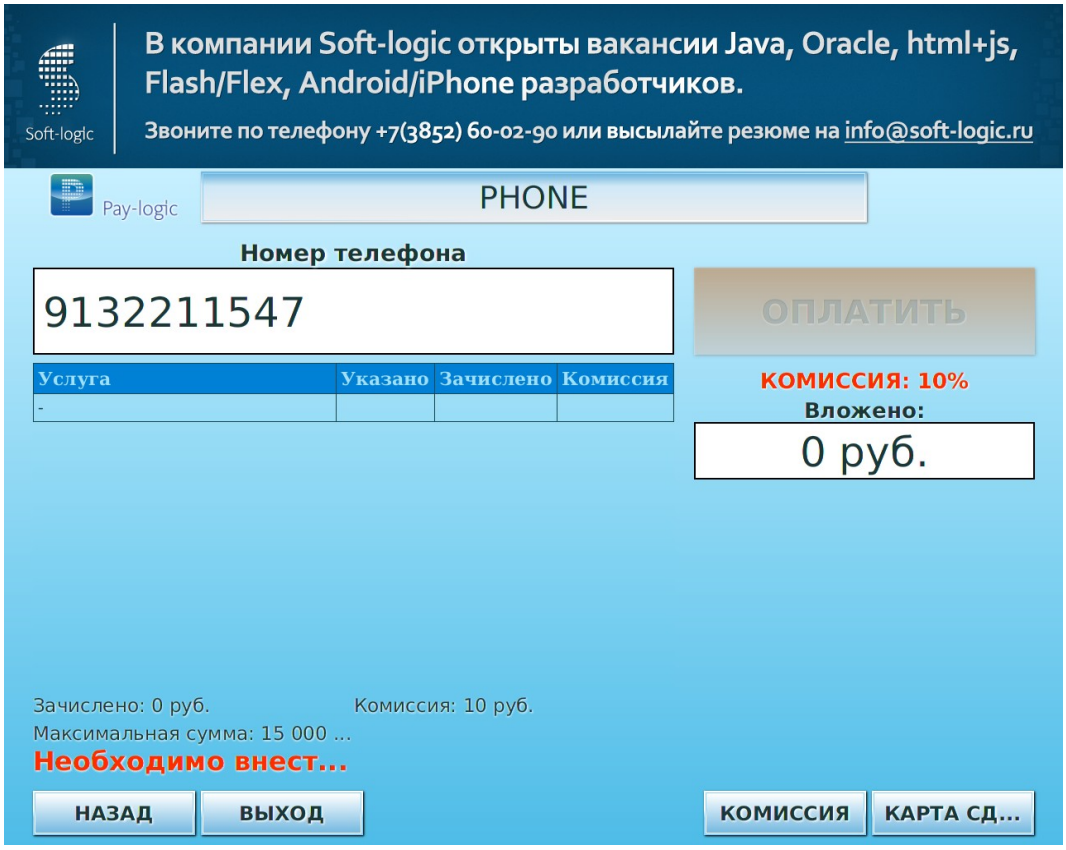


Рисунок 4.6.41.2 — Использование альтернативного экрана оплаты CASH/COMMUNAL

В случае если в сценарии прописан экран, как в примере, и настроена рекламная кампания с местом «SUM» так, как написано в документе «Реклама. Программное обеспечение «Процессинговый центр Pay-logic». Руководство пользователя», то на экране оплаты будет отображаться рекламное место «SUM-BANNER» (рисунок 4.6.41.3).

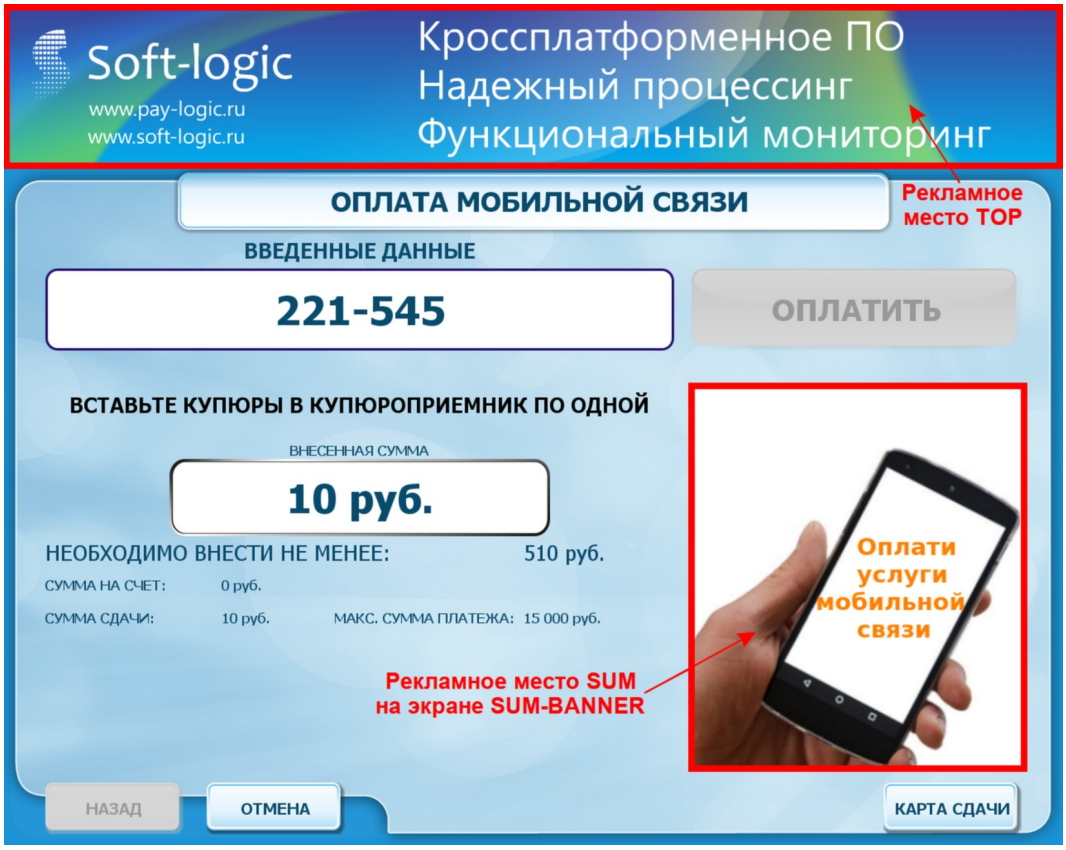


Рисунок 4.6.41.3 — Рекламное место с кодом SUM на экране `<screen type="Sum" sum="sum-banner"/>`

Пример:

```
<screen type="Sum" sum="cards-sum"/>
```

В случае, если в сценарии прописан экран, как в примере, и настроена рекламная кампания с местом «SUM-CARD» так, как написано в документе «Реклама. Программное обеспечение «Процессинговый центр Pay-logic». Руководство пользователя», то на экране оплаты будет отображаться рекламное место «SUM-CARD» (рисунок 4.6.41.4).

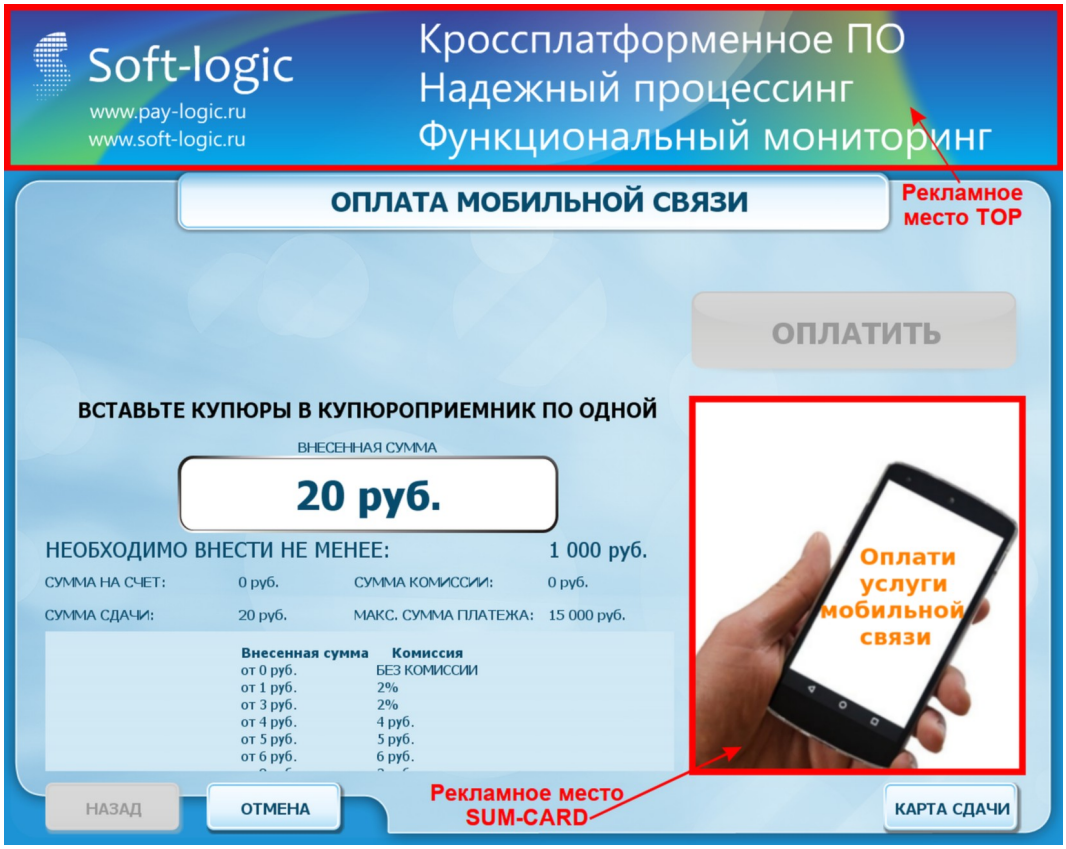


Рисунок 4.6.41.4 — Рекламное место с кодом SUM на экране `<screen type="Sum" sum="cards-sum"/>`

Доступен альтернативный экран оплаты sum-penalty (рисунок 4.6.41.5). На экране в верхнем поле отображается сумма к оплате с учетом комиссии и в поле ниже — внесенная в терминал сумма.

```
<screen type="sum" sum="sum-penalty"/>
```

Рисунок 4.6.41.5 — Альтернативный экран оплаты sum-penalty

4.6.42 ЭКРАН ВЫПОЛНЕНИЯ ДЕЙСТВИЙ (VOID)

Экран с типом **type=«void»** — экран предназначен для выполнения каких-либо фоновых задач, например, инициализации переменных перед переходом на другие экраны. Как правило, не отображается.

Пример:

```
<!--Создание экрана выполнения действий-->
<screen type="void" decor="button" title ="" id="first_screen">
  <fields/>
  <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="ДАЛЕЕ">
      <!--Объявление переменной sum-->
      <set key = "sum" key-title="Сумма" value="0" value-title="0.00"/>
        ...
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <goto-action type="Prev" title="НАЗАД" target="previous"/>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <goto-action type="Exit" title="ВЫХОД" target="exit"/>
  </actions>
</screen>
```

В данном примере при нажатии кнопки «Далее» переменной **sum** устанавливается значение **«0»**, после чего могут быть выполнены другие действия, при успешном завершении запроса происходит переход к экрану подтверждения.

4.6.43 ТАБЛИЧНЫЙ ЭКРАН (TABLE)

Экран с типом **`type=«table»`** предназначен для вывода таблицы на экран, рисунок 4.6.43.1. Для данного типа экрана доступно поле ввода `<table-field>` (раздел 5.10.2).

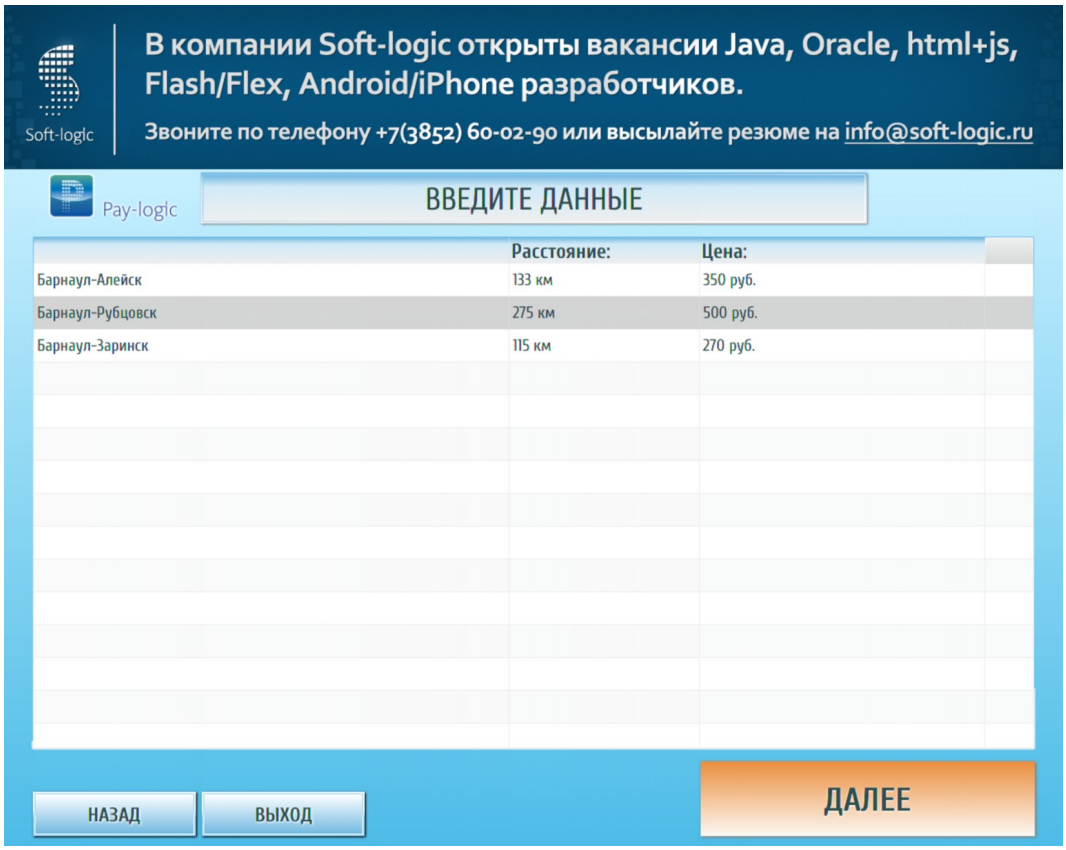


Рисунок 4.6.43.1 — Пример экрана TABLE для интерфейса Blues

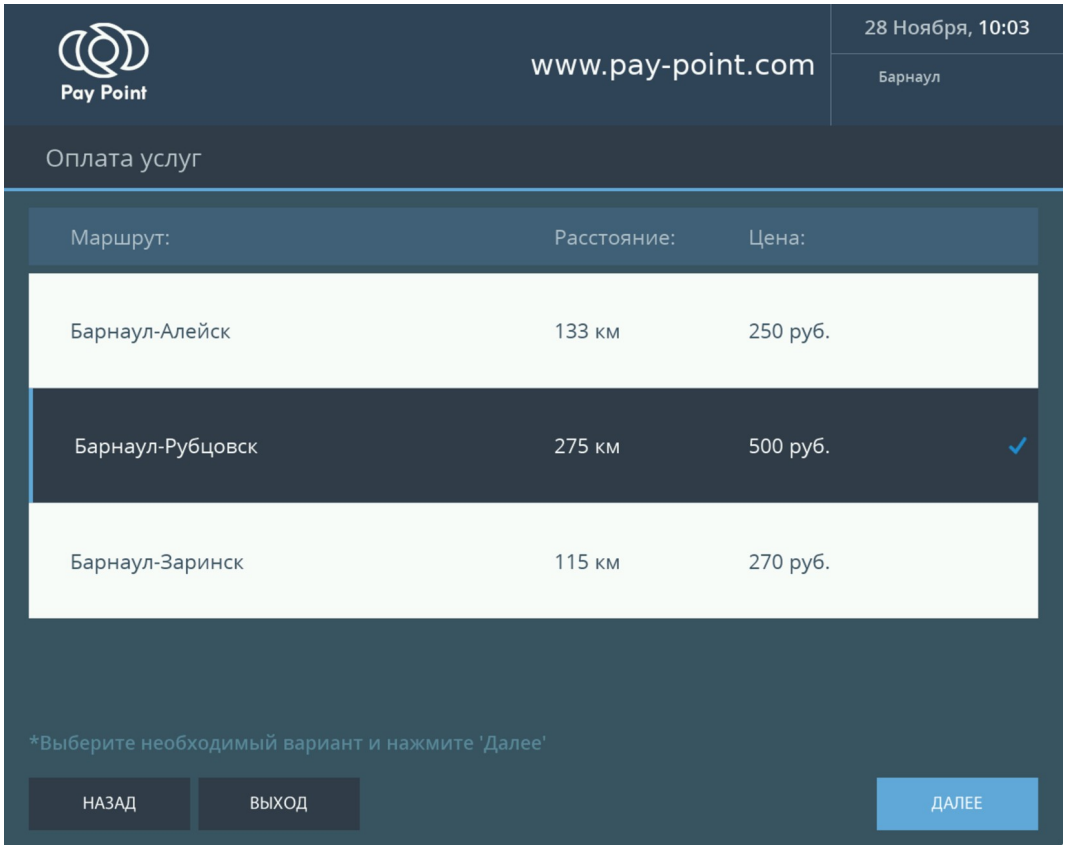


Рисунок 4.6.43.2 — Пример экрана TABLE для интерфейса Smoke

4.6.44 ТАБЛИЧНЫЙ ЭКРАН С НАВИГАЦИЕЙ (TABLE/NAVI)

Экран с типом **`type=«table/navi»`** — экран для вывода таблицы с навигацией по таблице на каждом экране. Доступен только в 7 версии ТПО. Навигация описана в разделе 4.4.

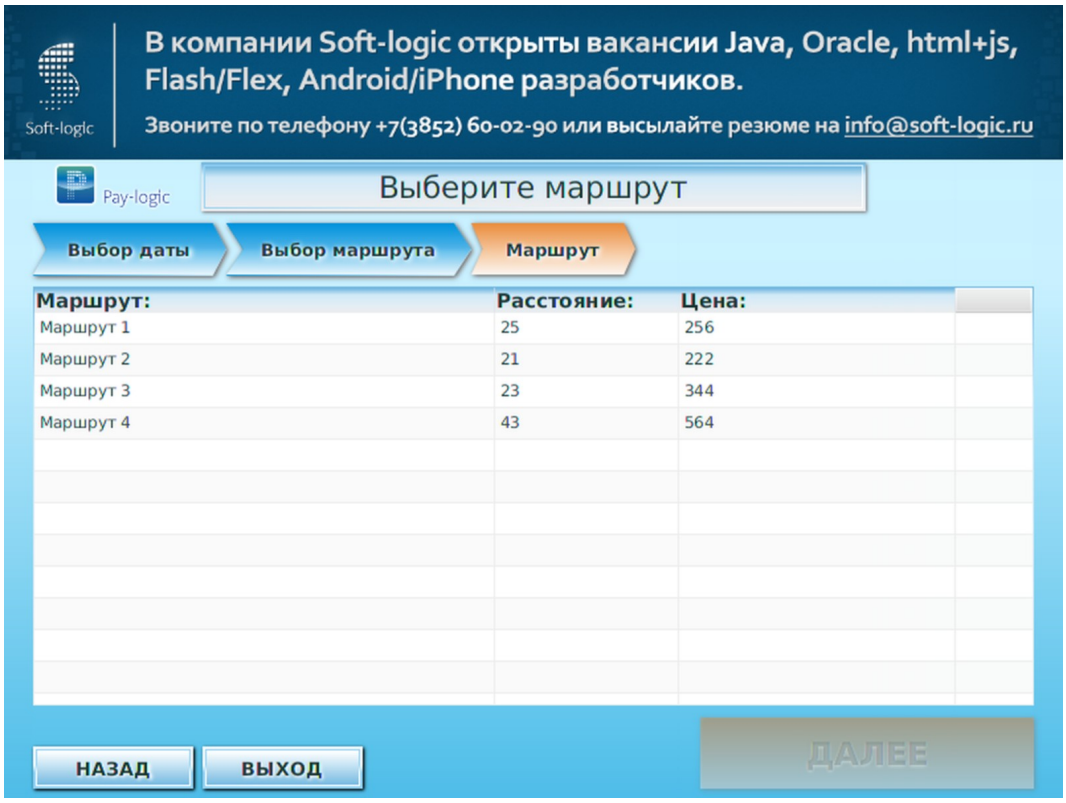


Рисунок 4.6.44.1 — Пример экрана TABLE/NAVI для интерфейса Blues

4.6.45 ЭКРАН ВЫБОРА МЕСТ В АВТОБУСЕ (BUSTICKETS)

Экран с типом **`type=«BusTickets»`** представляет собой графический селектор, который позволяет выбрать места в автобусе и оплатить билеты.

Рисунок 4.6.45.1 — Пример экрана BUSTICKETS для интерфейса BlueSphere2

В сценарии выбора и оплаты мест в автобусе возможно использовать разные шаблоны расположения мест в автобусе. Шаблоны описываются в файле `bustype.xml`, который должен располагаться в каталоге ***`<корень ТПО>/resources/utility/`***.

Пример:

```
<pre>
<!-- 6-17A -->
 <seat-layout name="6-17A" rows="3" places="6" row-distance="0.9"
              seat-distance="1.15" driver-icon="stering.png"
              seat-vacant="seat-open.png" seat-occur="seat-booking.png"
              seat-unavailable="seat-close.png">
  <seat-row id="1">
   <place id="A" type="driver"/>
   <place id="B" type="pass"/>
   <place id="C" type="seat" code="0.1"/>
   <place id="D" type="seat" code="0.2"/>
   <place id="E" type="seat" code="0.3"/>
   <place id="F" type="seat" code="0.4"/>
  </seat-row>
  <seat-row id="2">
   <place id="A" type="seat" code="1.0"/>
   <place id="B" type="seat" code="1.1"/>
   <place id="C" type="seat" code="1.2"/>
   <place id="D" type="pass"/>
   <place id="E" type="seat" code="1.3"/>
   <place id="F" type="seat" code="1.4"/>
  </seat-row>
  <seat-row id="3">
   <place id="A" type="seat" code="2.0"/>
   <place id="B" type="seat" code="2.1"/>
   <place id="C" type="seat" code="2.2"/>
   <place id="D" type="pass"/>
   <place id="E" type="seat" code="2.3"/>
   <place id="F" type="seat" code="2.4"/>
  </seat-row>
  <seat-row id="4">
   <place id="A" type="seat" code="3.0"/>
   <place id="B" type="seat" code="3.1"/>
   <place id="C" type="seat" code="3.2"/>
   <place id="D" type="pass"/>
   <place id="E" type="seat" code="3.3"/>
   <place id="F" type="seat" code="3.4"/>
  </seat-row>
  <seat-row id="5">
   <place id="A" type="seat" code="4.0"/>
   <place id="B" type="seat" code="4.1"/>
```

```
   <place id="C" type="seat" code="4.2"/>
   <place id="D" type="pass"/>
   <place id="E" type="seat" code="4.3"/>
   <place id="F" type="seat" code="4.4"/>
  </seat-row>
  <seat-row id="6">
   <place id="A" type="seat" code="5.0"/>
   <place id="B" type="seat" code="5.1"/>
   <place id="C" type="seat" code="5.2"/>
   <place id="D" type="pass"/>
   <place id="E" type="seat" code="5.3"/>
   <place id="F" type="seat" code="5.4"/>
  </seat-row>
 </seat-layout>
</pre>
```

Формат файла:

1. `<seat-layout>` — описывает тип автобуса, анализируется шлюзом и возвращается в ответе на запрос списка мест в автобусе. Параметр отвечает за отображаемый шаблон автобуса (макет мест) на экране;

   1) `<seat-row>` — описывает ряд в салоне автобуса:

      a) **`id`** — идентификатор, порядковый номер ряда;

      b) `<place>` – описывает место в автобусе:

      - **`id`** — идентификатор места;
      - **`type`** — определяет тип места. Возможные значения:
        - **`seat`** — пассажирское сиденье;
        - **`driver`** — водительское место;
        - **`pass`** — пустое место, используется, например, для изображения прохода в автобусе;
      - **`code`** — координаты места в формате, определенном реализацией. Например, «ряд.место».

Внимание!
В качестве разделителя между координатами места нельзя использовать двоеточие «:».

В каталоге ***<корень ТПО>/img/<интерфейс>/buttons/bus*** должны располагаться файлы с изображениями мест в салоне автобуса:

— место свободно;

— выбранное место;

— место недоступно;

 — место водителя.

Рассмотрим пример экрана, используемого для выбора мест.

Список мест возвращается шлюзом в корзине #places. Работа с корзиной платежей, подробно описана в разделе 7. В инпут-элементах корзины для идентификации места используется параметр **${place-id}**, возвращаемый шлюзом. Параметр используется для идентификации мест при формировании списка выбранных мест: `<pack type="data" key="#selected" elements="selected,summ,place-id" />`.

Пример:

```
<screen type="BusTickets" title="Выберите место" id="select_places">
 <!--Отображение полей из объекта NestedData #places-->
 <fields key="#places" />
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Next">
    <!--Подготовка атрибутов платежа-->
    <communal-prepare input-data="#places" filter="selected=1">
     <elements>
      <input-element key="place-id_${real-data-count}"
                     key-title="Идентификатор места" value="${place-id}"
                     value-title="${place-id}" original-value="${place-id}"/>
     </elements>
    </communal-prepare>
    <goto target="input_passenger_info" />
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <!--Удаление переменных из контекста-->
    <clear-like regex="^layout.*" />
    <goto target="select_round" />
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
    <goto target="exit" />
   </action>
   <action type="Edit" title="">
    <!--Если #selected не равно null-→
    <if condition="is-not-null #selected">
     <then>
      <!--Распаковка набора элементов-->
      <unpack type="data" key="#selected" elements="selected,place-id" />
      <!--Объявление переменной selected-->
      <set key="selected" value="1" />
      <!--Выполнение математических действий-->
      <math operation="ticket-count + 1" result-key="ticket-count"
            result-title="Total count" />
      <!--Упаковка набора элементов-->
      <pack type="data" key="#selected" elements="selected,summ,place-id" />
      <!--Удаление переменных из контекста-->
      <clear keys="#selected,selected" />
```

```
     <goto target="select_places" />
    </then>
    <else>
     <!--Иначе: если #removed не равно null-->
     <if condition="is-not-null #removed">
      <then>
       <!--Распаковка набора элементов-->
       <unpack type="data" key="#removed" elements="name" />
       <!--Отображение диалога с информационным сообщением-->
       <dialog type="Info" title="Удаление места"
               message="Вы уверены, что хотите удалить место {0}?"
               timeout="10" default="cancel" mess-params="place-number">
        <actions>
         <!--Описание поведения сценария, если нажата кнопка "Yes"-->
         <action type="okay" title="Yes">
          <!--Объявление переменной selected-->
          <set key="selected" value="0" value-title="0" />
          <!--Выполнение математических действий-->
          <math operation="ticket-count - 1" result-key="ticket-count"
                result-title="Total count" />
          <!--Упаковка набора элементов-->
          <pack type="data" key="#removed" elements="selected" />
          <!--Удаление переменных из контекста-->
          <clear keys="#removed,name" />
          <goto target="select_places" />
         </action>
         <action type="cancel" title="No">
          <!--Удаление переменных из контекста-->
          <clear keys="#removed" />
          <cancel/>
         </action>
        </actions>
       </dialog>
      </then>
     <else/>
    </if>
   </else>
  </if>
 </action>
</actions>
</screen>
```

4.6.46 ЭКРАН КИНОТЕАТРА (CINEMATICKETS)

Экран с типом **type=«CinemaTickets»** представляет собой графический селектор, который позволяет выбрать места в кинотеатре. Данный тип экрана доступен всем клиентам, использующим интерфейс BlueSphere2, для версий ТПО 5.

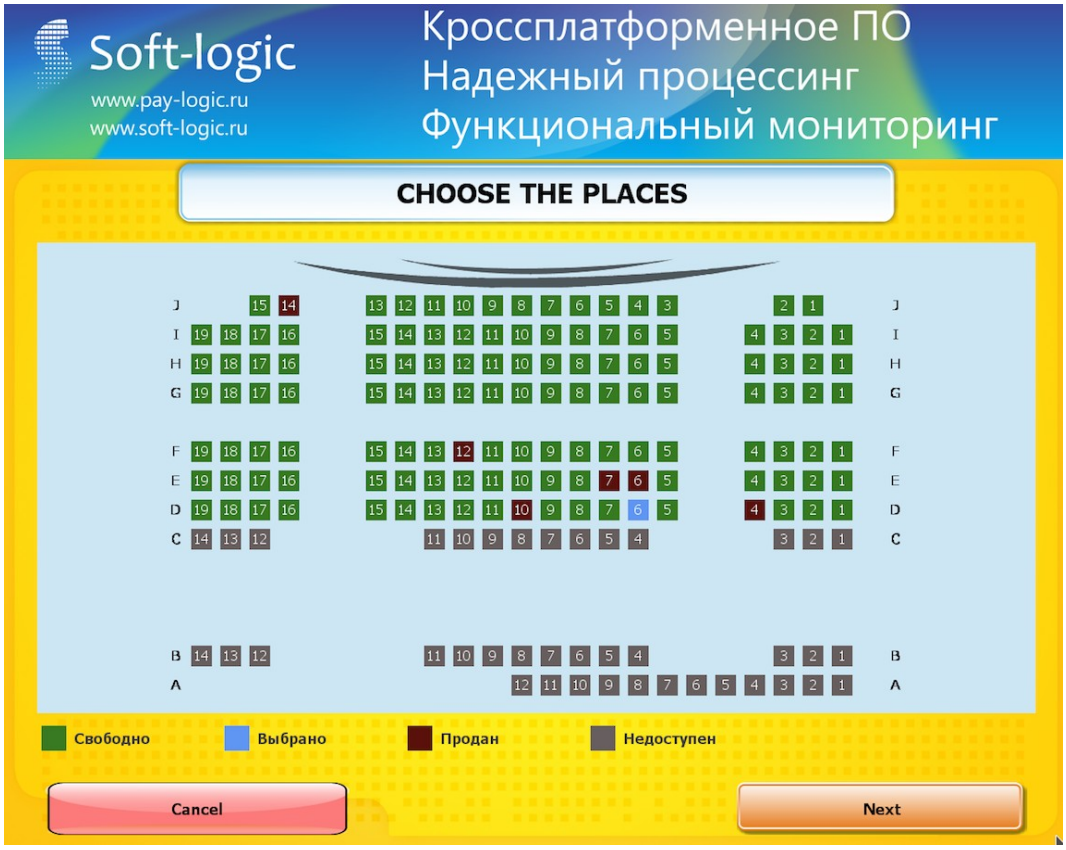


Рисунок 4.6.46.1 — Экран кинотеатра (BlueSphere2)

В сценарии выбора и оплаты мест в кинотеатре возможно использовать различные шаблоны расположения мест в кинотеатре. Шаблоны описываются в файле `cinematype.xml`, который располагается в каталоге ***<корень ТПО>/resources/utility/***.

Пример файла *cinematype.xml*, который описывает экран выбора мест на рисунке 4.6.46.1:

```
<seat-layout name="left-to-right_top-to-bottom _10_24_Silver" rows="10"
 places="24" rows-reverse="true" places-reverse="true"
 indent="5" screen-location="bottom" screen-image="screen_bottom.png">
    <seat-row id="1"
     places=""/>
    <seat-row id="2"
     places=""/>
    <pass-row id="3" row-spacing="2"/>
    <seat-row id="3"
     places="n;1;2;3;n;n;n;n;4;5;6;7;8;9;10;11;n;n;n;n;n;12;23;14"/>
    <seat-row id="4"
     places="n;1;2;3;4;n;n;5;6;7;8;9;10;11;12;13;14;15;n;n;16;17;18;19"/>
    <seat-row id="5"
     places="n;1;2;3;4;n;n;5;6;7;8;9;10;11;12;13;14;15;n;n;16;17;18;19"/>
    <seat-row id="6"
     places="n;1;2;3;4;n;n;5;6;7;8;9;10;11;12;13;14;15;n;n;16;17;18;19"/>
    <pass-row id="7" row-spacing="1"/>
    <seat-row id="7"
     places="n;1;2;3;4;n;n;5;6;7;8;9;10;11;12;13;14;15;n;n;16;17;18;19"/>
    <seat-row id="8"
     places="n;1;2;3;4;n;n;5;6;7;8;9;10;11;12;13;14;15;n;n;16;17;18;19"/>
    <seat-row id="9"
     places="n;1;2;3;4;n;n;5;6;7;8;9;10;11;12;13;14;15;n;n;16;17;18;19"/>
    <seat-row id="10"
     places="n;n;1;2;n;n;n;3;4;5;6;7;8;9;10;11;12;13;n;n;14;15;n;n"/>
    <row-symbols symbols="A;B;C;D;E;F;G;H;I;J"/>
</seat-layout>
```

Перед тем, как перейти на экран выбора мест в кинотеатре, клиент выбирает зону, в которой он желает приобрести билет (для выбора зоны кинотеатра используется другой экран, он не описывается в данном разделе). Выбранная зона будет называться «активной зоной». В данном примере активная зона имеет название «left-to-right_top-to-bottom _10_24_Silver» и занимает первые два ряда (id=«1», «2»). Информация о местах для активной зоны анализируется шлюзом и возвращается в ответе на запрос списка мест в кинотеатре, поэтому синтаксис задания мест для активной зоны отличается. Все остальные ряды (id=«3» - «10») находятся не в активной зоне, поэтому номера и количество мест в данных рядах необходимо явно задать в xml-файле, как показано в примере выше.

Структура файла *cinematype.xml*:

1. <seat-layout> – описывает зону кинотеатра. Значения атрибутов данного элемента возвращаются в ответе на запрос списка мест в кинотеатре. Отвечает за отображаемый шаблон зоны кинотеатра на экране. Имеет следующие атрибуты:

   1) **name** — наименование зоны кинотеатра;

   2) **rows** — количество рядов в зоне кинотеатра;

   3) **places** — количество мест в ряду;

   4) **rows-reverse** — имеет значение «false», если отсчет рядов ведется слева направо и «true» в противном случае;

   5) **places-reverse** — имеет значение «false», если отсчет мест ведется слева направо и «true» в противном случае;

   6) **indent** — отступ в пикселях между местами при отображении на экране кинотеатра. Значение по умолчанию «5»;

   7) **screen-location** — расположение экрана относительно рядов кинотеатра. Возможные значения атрибута:

      a) **top** — экран располагается вверху экрана;

      b) **bottom** — экран располагается внизу экрана.

   8) **sreen-image** — файл с изображением экрана кинотеатра. Его необходимо выложить в директорию: ***atm5/img/bluesphere2/buttons/cinema/***.

   9) <seat-row> — описывает ряд в кинотеатре. Возможно задать атрибуты:

      a) **id** — идентификатор, порядковый номер ряда;

      b) **places** — описывает номер места. Если установлено значение **«n»**, то данное место не отображается на экране кинотеатра.

10) `<row-symbols>` — описывает обозначение рядов кинотеатра. Имеет следующие атрибуты:

a) **`symbols`** — наименование ряда. В качестве значений атрибута могут использоваться цифры или буквы. По умолчанию нумерация рядов происходит с символа «1». При явном указании числа или буквы нумерация начнется с этого числа (буквы). Например:

- при указании числа «2», ряды будут иметь номера 2, 3, 4 и т.д.
- если задать латинскую «K», то ряды будут обозначены как K, L, M и т.д.
- при указании русской «Б» ряды будут обозначены как Б, В, Г, Д и т.д.

11) `<pass-row>` — служит для задания отступа между зонами. Включает следующие атрибуты:

a) **id** — идентификатор ряда, перед которым необходимо задать отступ;

b) **row-spacing** — количество «невидимых мест», на которое задается отступ.

5 ЭЛЕМЕНТЫ ВВОДА

5.1 ОБЩАЯ ИНФОРМАЦИЯ

Элемент `<fields>` внутри экрана описывает список полей ввода/вывода информации. Структура элемента:

```
<fields>
      ...
</fields>
```

В качестве полей ввода могут быть указаны элементы, приведенные в таблице 5.1.1.

Таблица 5.1.1 — Поля ввода

Поле ввода	Элемент
Текстовое поле ввода	`<text-field>` (раздел 5.7)
Числовое поле ввода	`<numeric-field>` (раздел 5.8)
Флаг/переключатель	`<checkbox-field>` (раздел 5.9)
Простой селектор	`<selector-field>` (раздел 5.10.1)
Табличный селектор	`<table-field>` (раздел 5.10.2)
Поле выбора даты	`<date-field>` (раздел 5.11)
Поле с автозаполнением	`<autocomplete-field>` (раздел 5.12)
Поле ввода и расшифровки информации, считанной со штрих-кода	`<barcode-field>` (используется только в 5 версии ТПО)

5.2 ОБЩИЕ АТРИБУТЫ ЭЛЕМЕНТОВ ВВОДА

Перечень атрибутов, общих для всех элементов ввода, приведен в таблице 5.2.1.

Таблица 5.2.1 — Общие атрибуты полей ввода

Атрибут	Описание	Обяз.
id	Ключ атрибута платежа	**Да**
group-id	Значение, по которому происходит группировка полей на экране, то есть поля, для которых указано одно и то же значение **group-id**, визуально будут объединены на экране в один блок. Поддерживается некоторыми интерфейсами	Нет
title	Наименование атрибута платежа, в кабинете отображается в окне информации о платеже в блоке «Общая информация», также может добавляться в чек	Нет
message	Заголовок экрана ввода	Нет
on-top	Позволяет редактировать значение поля ввода во всплывающем окне	Нет
width	Ширина данного поля в текущей строке в процентах, если используется конструкция <row>. Если сумма значений **width** всех полей внутри <row> превышает 100, то возникнет ошибка при обработке сценария. Если сумма значений **width** всех полей внутри <row> превышает 100, но **width** указан не для всех полей, то возникнет ошибка при обработке сценария. Если значение не указано, то считается, что такое поле занимает все свободное	Нет

Атрибут	Описание	Обяз.
	место. Если таких полей несколько, то они делят все свободное место поровну. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper».	
rowId	Идентификатор строки, в которой будет расположено поле. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	Нет

В этом примере при редактировании поля ввода номера договора будет отображаться всплывающее окно (рисунок 5.2.1).

Пример:

```
<scenario begin="1screen">
 <!--Создание экрана ввода данных с несколькими полями-->
 <screen type="group" title ="Введите данные" id="1screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 11. Название поля
ввода: "Номер договора"-->
   <text-field id="seria" title="Номер договора" max-len="11"
               keyboard="Digital" on-top="group/popup">
    <!--Регулярное выражение для валидации номера договора-->
    <validator type="regex">
     <rules>
      <rule regex="^\d{1,11}$"/>
     </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите номер документа]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 13. Название поля
ввода: "Номер служебного телефона"-->
```

```
    <text-field id="nomer" title="Номер служебного телефона"
                   max-len="13" keyboard="Digital">
      <!--Регулярное выражение для валидации номера договора-→
      <validator type="regex">
        <rules> <rule regex="^\d{10}$"/> </rules>
      </validator>
      <!--Регулярное выражение для форматирования вводимого номера на
экране-->
      <formatter type="regex">
        <rules default="8(***)***-**-**"/>
      </formatter>
      <!--Подсказка, отображается на экране-->
      <help> Введите номер служебного телефона </help>
    </text-field>
   </fields>
   <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <goto-action type="Next" title="Далее" target="pay"/>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <goto-action type="Prev" title="Назад" target="previous"/>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <goto-action type="Exit" title="Выход" target="exit"/>
   </actions>
  </screen>
</scenario>
```

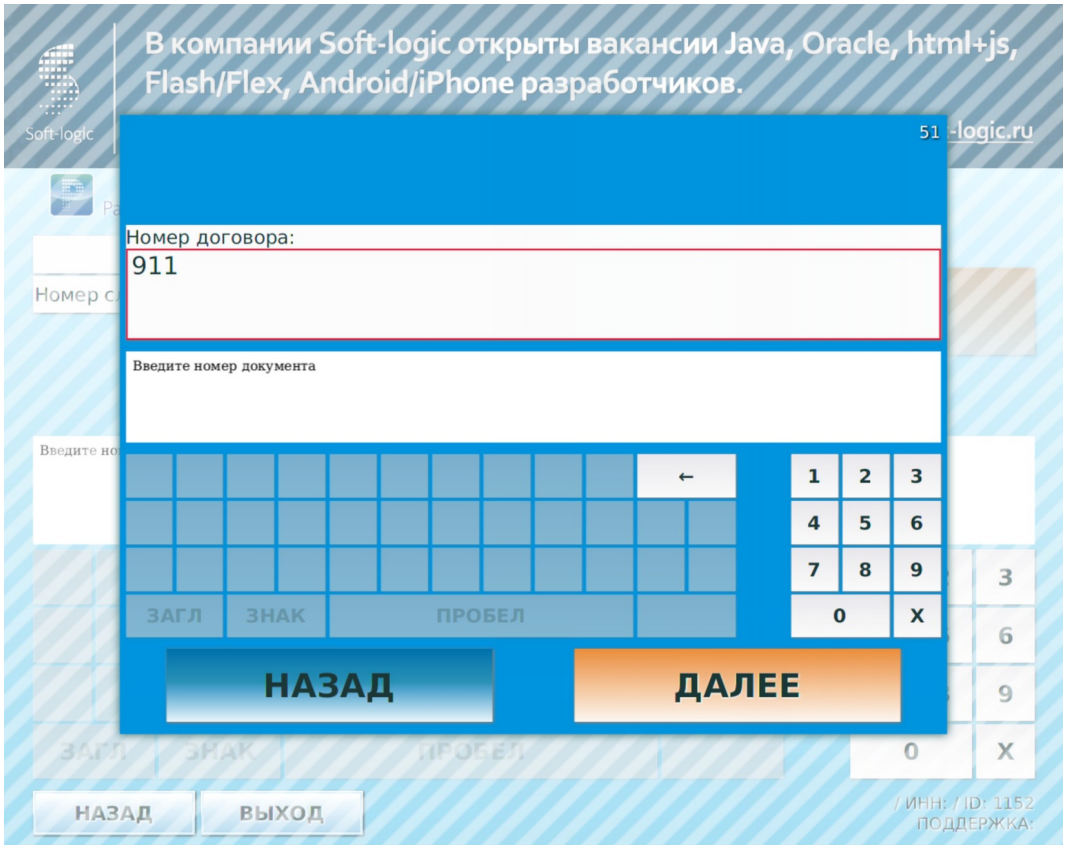


Рисунок 5.2.1 — Редактирование значения поля ввода во всплывающем окне

5.3 МЕХАНИЗМЫ ЛОКАЛИЗАЦИИ СЦЕНАРИЕВ

Существуют два механизма локализации сценариев. Первый предполагает создавать для каждого языка отдельный файл сценария с кодом локали, например, *110_ru.xml*, *110_en.xml* и т. д. Язык подгружаемого сценария сопоставляется с выбранным языком пользователя.

Другой механизм локализации предполагает выносить коды текстовок в отдельный файл. Ресурсные файлы с текстовками в ТПО 7 версии расположены в каталоге ***<корень ТПО>/res/module/i18n/*** в файлах *input.properties*, *input_<код локали>.properties*, в ТПО 5 версии — ***<корень ТПО>/resources/i18n/*** в файлах *input.properties*, *input_<код локали>.properties*. Код текстовки для того или иного языка указывается с помощью определенных атрибутов. Замена осуществляется автоматически при смене пользователем языка интерфейса.

Атрибуты, используемые для интернационализации, приведены в таблице 5.3.1, а также в описании элементов сценариев в соответствующих разделах (такие атрибуты содержат в названии «**-id**»).

Таблица 5.3.1 — Атрибуты, используемые для интернационализации

Атрибут	Описание	Обяз.
title-id	Содержит идентификатор текстовки, которая будет подгружена в качестве значения **title**	Да
message-id	Идентификатор текстовки, которая будет использована в качестве сообщения заголовка экрана ввода	Нет

5.4 ДВУХУРОВНЕВАЯ ЗАГРУЗКА СЦЕНАРИЕВ

Для повышения скорости обработки и загрузки сценариев предусмотрен функционал двухуровневой загрузки Рекомендуется использовать для сценариев объем кода, которых превышает 1000 строк.

Для реализации двухуровневой загрузки файл сценария разбивается на два:

1. Первый файл — основной и содержит все настройки для сценария и первый экран или несколько начальных экранов (например, если сценарий начинается с экрана «Void»). Имя основного сценария стандартное, например 123.xml.

2. Второй файл, содержит все остальные экраны, которые будут создаваться во время показа первой части. Имя дополнительного сценария состоит из имени первого файла и постфикса «_slave», то есть, например, 123_slave.xml.

Предупреждение!
Все опции указанные в дополнительном сценарии будут проигнорированы.

Использование двухуровневой загрузки позволяет существенно снизить время обработки сценария. Например, для сценария время обработки которого первоначально составляло ~150 мс, после разбиения первая часть обрабатывалась за ~20 мс, а обработку второй части пользователь не замечал.

Пример (первый файл):

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!-- E-Finance  -->
 <!--Создание экрана выбора-->
 <screen type="selector" title="Тип оплаты" id="0screen">
  <fields>
   <!--Создание поля выбора типа оплаты-->
   <selector-field id="id2" title="Тип оплаты">
    <items type="static">
     <item value="1" title="Погашение займа клиентом"/>
```

```
    <item value="2" title="Платёж агента по одному договору"/>
    <item value="3" title="Платёж агента по реестру договоров"/>
   </items>
  </selector-field>
 </fields>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии "Далее"-->
  <action type="Next" title="ДАЛЕЕ">
   <!--Если значение переменной id2 равно 1-->
   <if condition = "id2 == 1">
    <!--Переход на экран с id 01screen-->
    <then> <goto target="01screen"/> </then>
    <else/>
   </if>
   <!--Иначе: если значение переменной id2 равно 2-->
   <if condition = "id2 == 2">
    <!--Переход на экран с id 02screen-->
    <then> <goto target="02screen"/> </then>
    <else/>
   </if>
   <!--Иначе: если значение переменной id2 равно 3-->
   <if condition = "id2 == 3">
    <!--Переход на экран с id 03screen-→
    <then> <goto target="03screen"/> </then>
    <else/>
   </if>
  </action>
  <!--Описание поведения сценария оплаты при нажатии "Назад"-->
  <action type="Prev" title="Назад">
   <goto target="previous"/>
  </action>
  <!--Описание поведения сценария оплаты при нажатии "Выход"-->
  <action type="Exit" title="ВЫХОД">
   <goto target="exit"/>
  </action>
 </actions>
</screen>
</scenario>
```

Пример (второй файл):

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!-- 01screen -->
 <!--Создание экрана ввода числовой и текстовой информации-->
 <screen type="numeric" title="Введите № договора" id="01screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 11. Название поля
ввода: "№ договора"-->
   <text-field id="id1" title="№ договора" max-len="11"
               keyboard="Digital">
    <!--Регулярное выражение для валидации номера договора-->
    <validator type="regex">
     <rules> <rule regex="^\d{11}$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого номера на
экране-->
    <formatter type="regex">
     <rules default="**** *****"/>
    </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите № договора </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <goto target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="0screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход"> <goto target="exit"/> </action>
  </actions>
 </screen>
 <!-- 02screen -->
 <!--Создание экрана ввода данных с несколькими полями-->
 <screen type="group" title="Введите № договора,телефона" id="02screen">
  <fields>
```

```
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 11. Название поля
ввода: "№ договора"-->
   <text-field id="id1" title="№ договора" max-len="11"
               keyboard="Digital">
    <!--Регулярное выражение для валидации номера договора-->
    <validator type="regex">
     <rules> <rule regex="^\d{11}$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого номера на
экране-->
    <formatter type="regex">
     <rules default="** *** *** ***"/>
    </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите № договора </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура с возможностью переключения языка,
максимальное количество вводимых символов 10. Название поля ввода: "№
телефона"-->
   <text-field id="ph" title="№ телефона" max-len="10"
               keyboard="Digital">
    <!--Регулярное выражение для валидации номера телефона-->
    <validator type="regex">
     <rules> <rule regex="^\d{10}$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого номера на
экране-→
    <formatter type="regex">
     <rules default="8(***)***-**-**"/>
    </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите № служебного телефона </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <!--Объявление переменной selected-->
    <set key="sl" value="/" value-title="/"/>
    <str operation="id1 + sl" result-key="id1"
         result-title="№дог-ра,№тел."/>
```

```
    <str operation="id1 + ph" result-key="id1"
         result-title="№дог-ра,№тел."/>
    <goto target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="0screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
 <!--Создание экрана ввода числовой и текстовой информации-->
 <screen type="numeric" title="Введите № телефона" id="03screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 10. Название поля
ввода: "№ телефона"-->
   <text-field id="id1" title="№ телефона" max-len="10"
               keyboard="Digital">
    <!--Регулярное выражение для валидации номера телефона-->
    <validator type="regex">
     <rules> <rule regex="^\d{10}$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого номера на
экране-->
    <formatter type="regex">
     <rules default="8(***)***-**-**"/>
    </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите № служебного телефона </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <goto target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="0screen"/>
```


Web: https://soft-logic.ru/
Mail: info@soft-logic.ru

```
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
</scenario>
```

5.5 ФЛАГИ

Флаги могут задаваться как числом, так и символьными константами. Может указываться одновременно несколько флагов через символ «|» (вертикальная черта). Доступные флаги приведены в таблице 5.5.1.

Таблица 5.5.1 — Флаги

Символьная константа	Числовое значение	Описание
HIDE_ON_CONFIRM	0x01	Не показывать на экране подтверждения. Доступен на ТПО и РМА

HIDE_ON_PRINT	0x02	Не выводить на печать в чеке. Используется, в том случае когда атрибут платежа, не нужно печатать на чеке. В шаблоне чека необходимо использовать цикл:
		`#foreach( $key in $operation.inputElements)`
		`  $key.keyTitle: $key.value`
		`#end`
		key — ключ атрибута.
		Доступен на ТПО и РМА

HIDE_ON_SERVER	0x04	Не показывать в атрибутах платежа в кабинете. Распространяется на атрибуты **id1**, **id2**. Администратору ПС атрибут отображается вне зависимости от значения флага. Доступен на ТПО и РМА

HELP_KEEPER_ANDROID	0x08	Подсказка для Android клиентов
HIDE_FROM_EXTERNAL	0x100	Запрет отдачи атрибута внешним агентам
HIDE_ON_FRONTEND	0x200	Не отображать атрибут плательщику. Используется, в том случае когда для совершения платежа передается служебный атрибут, который не нужно показывать пользователю. Доступен в электронном кошельке. Для роли пользователя «Администратор» атрибут не скрывается
USE_CONTACTS	0x10	Возможность использования контактов из телефонной книги. Используется на мобильных клиентах электронного кошелька
VALIDATE_BY_CAPACITY	0x20	Использовать валидацию по ёмкостям. Предназначен для полей, содержащие телефонные номера
HELP_MOBILE_ANDROID	0x40	Подсказка для терминалов Android
HELP_MOBILE_IOS	0x80	Подсказка для терминалов IOS
RECEIVED_FROM_SERVER	0x300	Атрибут был получен от сервера. Используется при онлайн-запросах. Для использования в коде сценария недоступен. Используется разработчиками
VALUE_IS_UNCHANGED	0x400	Позволяет помечать input-elementы, как нередактируемые
FLAG_MASK_IN_LOGS	0x800	Позволяет отправлять атрибуты в логах в маскированном виде. При этом **originalValue**,

		valueTitle не передаются, а в **value** каждый оригинальный символ значения заменяется на символ «*». На атрибуты id1, id2 не действует
FLAG_MASK_CARD_IN_LOGS	0x8000	Позволяет шифровать в логах номера карт по правилу ^(\d{6}|\d{4}\s\d{2}).*(\d{4})$ => $1******$2
FLAG_VERIFIED_PERSONAL_DATA	0x20000	Признак хранения в атрибуте персональных данных клиента. При использовании флага на РМА данные маскируются по умолчанию, в кабинете ПЦ данные не отображаются в карточке операции диспетчерской. На ТПО флаг не обрабатывается.
FLAG_DO_NOT_SAVE_TO_DB	0x40000	Атрибут не требует сохранения в БД. При сохранении результатов операции, атрибут с таким флагом будет удален из БД
FLAG_ATTRIBUTE_ENCRYPTED	0x40000000	Позволяет шифровать атрибуты на стороне сервера

Пример сценария с маскированием значения атрибута в логах (7 версия ТПО):

```
<scenario begin="0screen">
 <!--Создание экрана ввода числовой и текстовой информации-->
 <screen type="numeric" title ="Введите номер брони" id="0screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 6. Название поля ввода:
"Номер брони"-->
   <text-field id="id1" title="Номер брони" max-len="6" keyboard="Digital"
               flags="0x800">
    <!--Регулярное выражение для валидации номера брони-->
    <validator type="regex">
     <rules> <rule regex="^.{6}$"/> </rules>
```

```
      </validator>
      <!--Подсказка, отображается на экране-->
      <help> <![CDATA[Введите номер брони]]> </help>
    </text-field>
  </fields>
  <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="Далее">
      <goto target="1screen"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <action type="Prev" title="Назад">
      <goto target="previous"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <action type="Exit" title="Выход">
      <goto target="exit"/>
    </action>
  </actions>
 </screen>
 <!--Создание экрана ввода данных с несколькими полями-->
 <screen type="group" decor="simple" title ="Введите данные" id="1screen">
  <fields>
    <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский без возможности
переключения языка, максимальное количество вводимых символов 60. Название
поля ввода: "ФИО плательщика". В логах и на сервере атрибут отображается в
маскированном виде-->
    <text-field id="fio" title="ФИО плательщика" max-len="60"
                keyboard="any:[ru,symb]:upper:true"
flags="FLAG_MASK_IN_LOGS">
      <!--Регулярное выражение для валидации введенных данных-->
      <validator type="regex">
        <rules>
          <rule regex="^[a-яА-Яёё-]{2,20}[\s]{1}[a-яА-Яёё-]{2,20}[\s]{1,3}
                      [a-яА-ЯЁё-]{2,20}([\s]{1}[a-яА-ЯЁё-]{2,20})?$"/>
        </rules>
      </validator>
      <!--Подсказка, отображается на экране-->
      <help> <![CDATA[Введите ФИО плательщика]]> </help>
    </text-field>
    <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский без возможности
```

```
переключения языка, максимальное количество вводимых символов 254. Название
поля ввода: "Адрес плательщика"-->
    <text-field id="address" title="Адрес плательщика" max-len="254"
            keyboard="any:[ru,symb]:upper:true">
     <!--Регулярное выражение для валидации введенных данных-->
     <validator type="regex">
      <rules> <rule regex="^.{3,254}$"/> </rules>
     </validator>
     <!--Подсказка, отображается на экране-->
     <help> <![CDATA[Введите ФИО плательщика]]> </help>
    </text-field>
    <!--Создание поля выбора типа оплаты-->
    <selector-field id="type" title="Тип оплаты">
     <items type="static">
      <item value="21" title="Паспорт РФ"/>
      <item value="22" title="Заграничный паспорт"/>
      <item value="31" title="Паспорт иностранного гражданина"/>
     </items>
    </selector-field>
   </fields>
   <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="Далее">
     <goto target="pay"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <action type="Prev" title="Назад">
     <goto target="previous"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <action type="Exit" title="Выход">
     <goto target="exit"/>
    </action>
   </actions>
  </screen>
</scenario>
```

В маскированном виде будут переданы **`value`** атрибутов **`id1`**, **`fio`**.

Фрагмент network.log:

```
15:47:46,767  INFO [JavaFX Application Thread] Add operation:
Operation[service=Лотерея, inputElements={id1=InputElement{key=id1,
```

keyTitle=Номер брони, keyTitleId=null, value=******, flags=100000000000},
address=InputElement{key=address, keyTitle=Адрес плательщика,
keyTitleId=null, originalValue=УЛ. А. ПЕТРОВА 76-82, value=УЛ. А. ПЕТРОВА
76-82, valueTitle=УЛ. А. ПЕТРОВА 76-82, flags=0, alternatives=[]},
type=InputElement{key=type, keyTitle=Тип оплаты, keyTitleId=null,
originalValue=21, value=21, valueTitle=Паспорт РФ, flags=0,
alternatives=[]}, fio=InputElement{key=fio, keyTitle=ФИО плательщика,
keyTitleId=null, value=******************, flags=100000000000}},
form={id1=255665, address=УЛ. А. ПЕТРОВА 76-82, type=21, fio=ИВАНОВ ИВАН
ИВАНОВИЧ}, views={id1=255665, address=УЛ. А. ПЕТРОВА 76-82, type=Паспорт
РФ, fio=ИВАНОВ ИВАН ИВАНОВИЧ}, titles={id1=Номер брони, address=Адрес
плательщика, type=Тип оплаты, fio=ФИО плательщика}, id=1, check=31146,
pinCode=null, pinCodeNominal=null, date=Wed Mar 22 15:47:08 NOVT 2017,
change-date=null, fraudControlRule=null]

5.6 КЛАВИАТУРЫ

5.6.1 КЛАВИАТУРЫ ТПО

В ТПО 5, 7 версий предусмотрены следующие типы клавиатур:

1. Цифровая клавиатура — позволяет вводить пользователю только цифры, и возможно — десятичный разделитель. Цифровая клавиатура задается кодом Digital (рисунок 5.6.1.1).

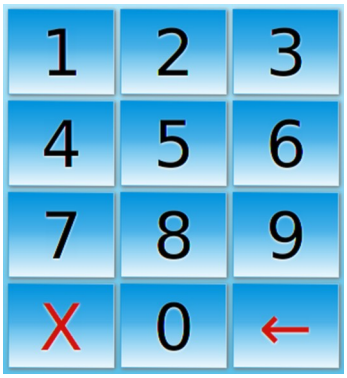


Рисунок 5.6.1.1 — Digital-клавиатура

Вместо функциональной кнопки «X» (очистить) возможно указать символ из символьной или буквенной раскладки клавиатуры. Например, отображение клавиатуры, с косой чертой вместо символа «X» (очистить) приведено на рисунке 5.6.1.2.

2. Полноценная (полноразмерная клавиатура). Внешний вид клавиатуры определяется опциями (any|letter|digital):(\[([a-z]{1,5},)*[a-z]{1,5}\])(:(lower|upper):(true|false)), которые позволяют настраивать отображение различных символов на клавиатуре, язык, регистр букв, возможность смены регистра:

1) часть (any|letter|digital) задает используемые символы, возможно использовать только одно из значений:

a) **`any`** — все символы;

b) **`letter`** — буквы и цифры;

c) **`digital`** — только цифры.

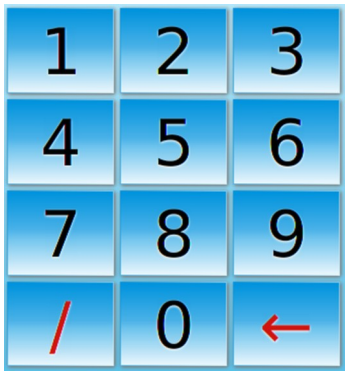


Рисунок 5.6.1.2 — Тип клавиатуры Digital с заменённой функциональной кнопкой «Стереть»

2) часть (\[([a-z]{1,5},)*[a-z]{1,5}\]) задает алфавит, наличие переключателя на символьную раскладку, возможность использования только цифровой раскладки на групповом экране:

a) **`ru`** — русский алфавит, **`en`** — английский алфавит и т. п.;

b) **`symb`** — наличие переключателя на символьную раскладку.

3) часть (lower|upper|mixed) задает регистр букв в раскладке по умолчанию (используется с letter раскладкой). Возможно использовать только одно из значений:

a) **`lower`** — нижний регистр;

b) **`upper`** — верхний регистр;

c) в ТПО 5 для интерфейса Smoke добавлен вариант клавиатуры «mixed» — смешанный регистр. Первая буква и каждая буква после пробела прописные, остальные — строчные. По запросу может быть внедрена поддержка типа клавиатуры в других типах интерфейсов.

4) часть (true|false) задает возможность смены регистра при вводе данных. Возможно использовать только одно из значений:

a) **`true`** — допустимо менять регистр символов при вводе;

b) **`false`** — невозможна смена регистра.

Части разделяются символом ":". Пример:

```
keyboard="any:[ru,symb]:lower:true"
```

Отображение клавиатуры, заданной записью «any:[ru,symb]:lower:true» приведено на рисунке 5.6.1.3.

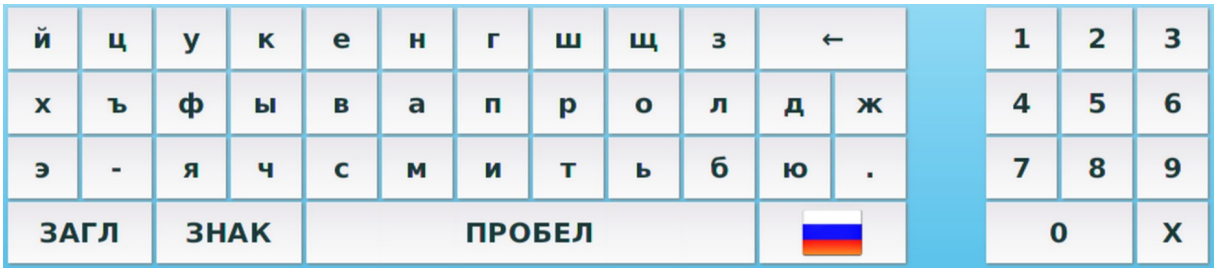


Рисунок 5.6.1.3 — Клавиатура any:[ru,symb]:lower:true

5.7 ТЕКСТОВОЕ ПОЛЕ ВВОДА (<TEXT-FIELD>)

Текстовое поле ввода описывается элементом `<text-field>`. На одном экране ввода может быть несколько текстовых полей ввода. Внешний вид элемента приведен на рисунке 5.7.1.

Рисунок 5.7.1 — Внешний вид элемента `<text-field>`

Структура элемента:

```
<text-field id=<tId>
            group-id=<nonNegativeInteger>
            title=<titleType>
            title-id=<resourceIdType>
            message=<messageType>
            message-id=<resourceIdType>
            on-top=<tId>
            flags=<flags_specification>
            keyboard=<keyboard_specification>
            xbutton=<notEmptyString1>
            letter-case=[UpperCase|LetterCase]
            secure=[true|false]
            default=<notEmptyString40>
            example=<notEmptyString40>
            error-message=<notEmptyString40>
            prefix=<notEmptyString10>
            postfix=<notEmptyString10>
            max-len=<positiveShort>
            read-only=[true|false]
            exist=[true|false]
            width=<nonNegativeInteger>
            rowId=<nonNegativeInteger>
            fail-regex=<string>
            trailing-widget=<string>
            next-regex=<string>>
      <validator> ... </validator>
```

```
        <data-formatter> ... </data-formatter>
        <formatter> ... </formatter>
        <help> ... </help>
        <modificator> ... </modificator>
        <filter> ... </filter>
</text-field>
```

Дочерние элементы: `<validator>` (раздел 5.14.1), `<data-formatter>` (раздел 5.14.2), `<formatter>` (раздел 5.14.3), `<help>` (раздел 5.13), `<modificator>` (раздел 5.14.4), `<filter>` (раздел 5.14.5).

Атрибуты описаны в таблице 5.7.1.

Таблица 5.7.1 — Атрибуты текстового поля ввода `<text-field>`

Атрибут	Описание	Обяз.
id	Ключ атрибута платежа	Да
group-id	Значение, по которому происходит группировка полей на экране, то есть поля, для которых указано одно и то же значение **group-id**, визуально будут объединены на экране в один блок. Поддерживается некоторыми интерфейсами	Нет
title	Наименование атрибута платежа, в кабинете отображается в окне информации о платеже в блоке «Общая информация», также может добавляться в чек	Нет
title-id	Содержит идентификатор текстовки, которая будет подгружена в качестве значения **title**	Нет
message	Заголовок экрана ввода. Поддерживается только на ТПО 7	Нет
message-id	Идентификатор текстовки, которая будет использована в качестве сообщения заголовка экрана ввода	Нет
on-top	Позволяет редактировать значение поля ввода во всплывающем окне	Нет

Атрибут	Описание	Обяз.
flags	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет
keyboard	Указывается тип клавиатуры, отображаемой на экране терминала. Типы клавиатур описаны в разделе 5.6	Да
xbutton	Указывается один символ из символьной или буквенной раскладки клавиатуры, который будет отображаться вместо функциональной кнопки «Стереть» (или сброс введенного значения) для цифровой раскладки. Данный атрибут используется только с цифровой раскладкой. При указании клавиатуры в 5 версии ТПО важен регистр (digital — неверно)	Нет
letter-case	Предоставляет возможность установить первую букву сообщения заглавной. Возможные значения: **UpperCase** — заглавная буква **LowerCase** — строчная буква	Нет
secure	Используется для возможности безопасного ввода данных, определены значения: • **true** — в поле ввода текстовой информации символы при вводе символы подменяются «*». Например, при вводе пароля; • **false** — в поле ввода текстовой информации символы при вводе символы не подменяются «*»	Нет
default	Используется для указания значения, выводимого в поле ввода по умолчанию, можно редактировать	Нет
example	Пример ввода информации, отображается на экране	Нет

Атрибут	Описание	Обяз.
error-message	Сообщение об ошибке для клиента, отображается на экране	Нет
prefix	Используется для указания префикса вводимых данных. Префикс не редактируется. Введённые данные будут переданы на сервер вместе с префиксом. Дополнительно в сценариях в сценариях в атрибуте **prefix** может быть использовано поле «Телефонный код», которое настраивается в разделе «Справочники — География — Страны». Например, **prefix="$phone_code"**. Телефонный код будет отображаться при вводе атрибута в сценарии перед вводимыми плательщиком данными	Нет
postfix	Используется для указания постфикса вводимых данных, не редактируется, добавляется к введённому значению, передается на сервер	Нет
max-len	Максимальное число вводимых символов, задает ограничение на длину поля	Да
read-only	Флаг «только чтение», принимает одно из двух значений: • **true** — только чтение, поле для ввода неактивно; • **false** — поле для ввода активно, используется по умолчанию. В административной части системы электронных кошельков SmartKeeper поддерживается с версии 3.1	Нет
exist	Определяет показывать поле только, если в контексте есть соответствующая переменная или вне зависимости от этого. Возможные значения: • **true** — поле отображается только, если в контексте есть соответствующая переменная; • **false** — поле отображается вне зависимости	Нет

Атрибут	Описание	Обяз.
	от того, есть ли в контексте соответствующая переменная	
width	Ширина данного поля в текущей строке в процентах, если используется конструкция <row>. Если сумма значений **width** всех полей внутри <row> превышает 100, то возникнет ошибка при обработке сценария. Если сумма значений **width** всех полей внутри <row> превышает 100, но **width** указан не для всех полей, то возникнет ошибка при обработке сценария. Если значение не указано, то считается, что такое поле занимает все свободное место. Если таких полей несколько, то они делят все свободное место поровну. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	Нет
rowId	Идентификатор строки, в которой будет расположено поле. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	Нет
fail-regex	Содержит регулярное выражение для проверки. Используется в тех случаях, когда нельзя гарантировать, что элемент <verify> валидирует все допустимые значения. Если введенные данные не удовлетворяют <verify>, но удовлетворяют **fail-regex**, то к атрибутам платежа добавляется элемент **Key.INVALID=«true»**. Если данный атрибут присутствует, то генерируется событие с типом «Неизвестный номер» (раздел кабинета «Мониторинг — События»), а в *pay.log* появляется запись «Invalid	Нет

Атрибут	Описание	Обяз.
	input data. Ask about payment». Для клиента процедура оплаты предполагает вывод диалогового окна с запросом подтверждения оплаты. Доступен в ТПО 5, 7	
next-regex	Если введенное в поле значение попадает под условия заданного в параметре регулярного выражения, то выполняется автопереход на следующий экран. Обработка параметра реализована только на экране NUMERIC в версиях 5.120 и выше в некоторых интерфейсах. Для других интерфейсов доработка осуществляется по запросу	Нет
trailing-widget	Добавление на экран специальных кнопок ввода данных. Например «Камера», «Список контактов». Поддержка функционала обеспечивается в рамках дизайна GUI.	Нет

Пример:

```
<!--Создание поля ввода, со следующими параметрами: активна цифровая и
буквенная клавиатура, язык ввода – русский без возможности переключения
языка, максимальное количество вводимых символов 60. Название поля ввода:
"ФИО плательщика"-->
<text-field id="fio" title="ФИО плательщика" max-len="60"
            keyboard="any:[ru]:lower:true">
   <!--Регулярное выражение для валидации введенных данных-->
   <validator type="regex">
      <rules>
         <rule regex="^[А-Яа-яЁё]+\s[А-Яа-яЁё]+\s[А-Яа-яЁё]+$"/>
      </rules>
   </validator>
   <!--Подсказка, отображается на экране-->
   <help>
      Введите Ваши ФИО
   </help>
</text-field>
```

Экран, соответствующий данному примеру, приведен на рисунке 5.7.2.

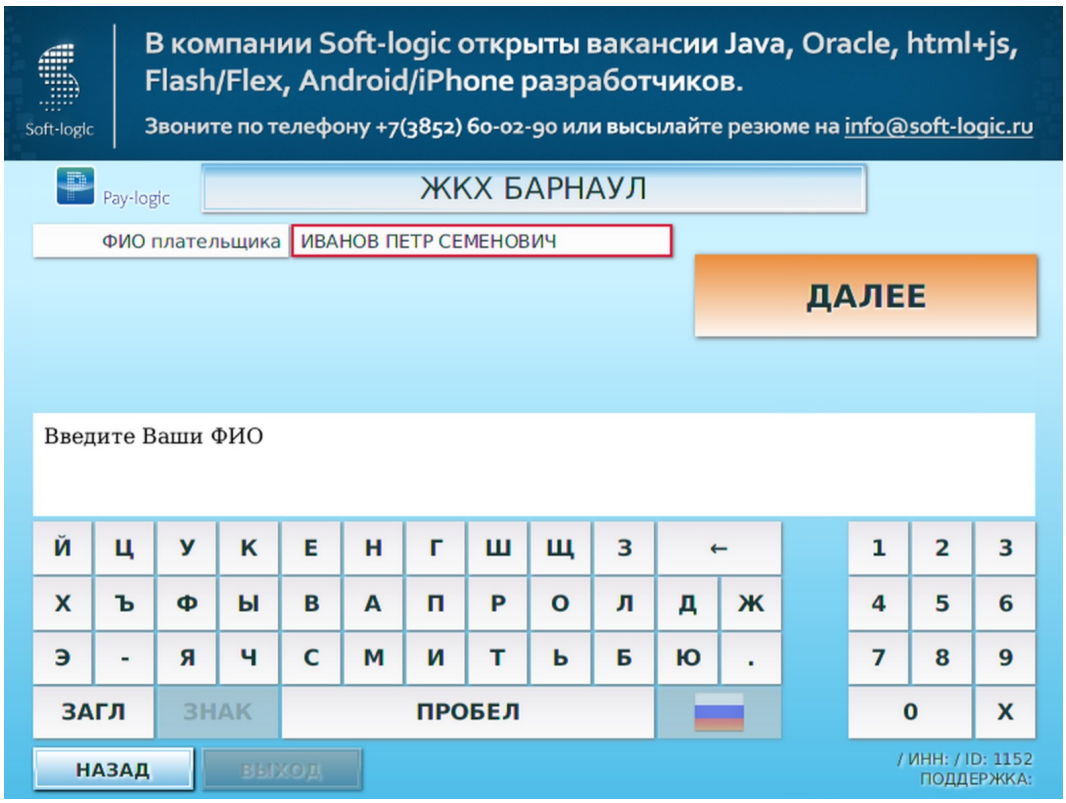


Рисунок 5.7.2 — Экран с элементом `<text-field>`

Пример использования next-regex:

```
<screen type="numeric" title ="Введите номер брони" id="0screen">
    <fields>
        <!--Создание поля ввода, со следующими параметрами: активна
цифровая клавиатура, максимальное количество вводимых символов 5.
Название поля ввода: "Номер брони"-->
        <text-field id="id1" title="Номер брони" max-len="5"
keyboard="Digital" next-regex="^.{5}$">
            <!--Регулярное выражение для валидации номера брони-->
            <validator type="regex">
                <rules> <rule regex="^.{5}$"/> </rules>
```

```
            </validator>
            <!--Подсказка, отображается на экране-->
            <help> <![CDATA[Введите номер брони]]> </help>
        </text-field>
    </fields>
    <actions>
        <!--Описание поведения сценария оплаты при нажатии кнопки
"Далее"-->
        <action type="Next" title="Далее">
            <goto target="1screen"/>
        </action>
        <!--Описание поведения сценария оплаты при нажатии кнопки
"Назад"-->
        <action type="Prev" title="Назад">
            <goto target="previous"/>
        </action>
        <!--Описание поведения сценария оплаты при нажатии кнопки
"Выход"-->
        <action type="Exit" title="Выход">
            <goto target="exit"/>
        </action>
    </actions>
</screen>
```

5.8 ЧИСЛОВОЕ ПОЛЕ ВВОДА (<NUMERIC-FIELD>)

Числовое поле ввода описывается элементом `<numeric-field>` и предназначено, как правило, для ввода суммы. Предусматривает десятичные значения. Внешний вид элемента приведен на рисунке 5.8.1.

Рисунок 5.8.1 — Внешний вид элемента `<numeric-field>`

Структура элемента:

```
<numeric-field id=<tId>
               group-id=<nonNegativeInteger>
               title=<titleType>
               title-id=<resourceIdType>
               message=<messageType>
               message-id=<resourceIdType>
               on-top=<tId>
               flags=<flags_specification>
               default=<notEmptyString40>
               example=<notEmprtyString40>
               error-message=<notEmprtyString40>
               read-only=[true|false]
               exist=[true|false]
               format=<string>
               width=<nonNegativeInteger>
               rowId=<nonNegativeInteger>>
      <help> ... </help>
      <verify> ...</verify>
</numeric-field>
```

Дочерние элементы: `<help>` (раздел 5.13), `<verify>`.

Атрибуты описаны в таблице 5.8.1.

Таблица 5.8.1 — Атрибуты числового поля ввода `<numeric-field>`

Атрибут	Описание	Обяз.
id	Ключ атрибута платежа	**Да**
group-id	Значение, по которому происходит группировка полей на экране, то есть поля, для которых указано одно и то же значение **group-id**, визуально будут объединены на экране в один блок. Поддерживается некоторыми интерфейсами	Нет
title	Наименование атрибута платежа, в кабинете отображается в окне информации о платеже в блоке «Общая информация», также может добавляться в чек	Нет
title-id	Содержит идентификатор текстовки, которая будет подгружена в качестве значения **title**	**Да**
message	Заголовок экрана ввода	Нет
message-id	Идентификатор текстовки, которая будет использована в качестве сообщения заголовка экрана ввода	Нет
on-top	Позволяет редактировать значение поля ввода во всплывающем окне	Нет
flags	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет
default	Используется для указания значения, выводимого в поле ввода по умолчанию, можно редактировать	Нет
example	Пример ввода информации, отображается на экране	Нет
error-message	Сообщение об ошибке для клиента, отображается на экране	Нет
read-only	Флаг «только чтение», принимает одно из двух значений:	Нет

Атрибут	Описание	Обяз.
	• **true** — только чтение, поле для ввода неактивно; • **false** — поле для ввода активно, используется по умолчанию. В административной части системы электронных кошельков «SmartKeeper» поддерживается с версии 3.1	
exist	Определяет показывать поле только, если в контексте есть соответствующая переменная или вне зависимости от этого. Возможные значения: • **true** — поле отображается только, если в контексте есть соответствующая переменная; • **false** — поле отображается вне зависимости от того, есть ли в контексте соответствующая переменная	Нет
format	Задается десятичный формат числа через точку, например 5.2	**Да**
width	Ширина данного поля в текущей строке в процентах, если используется конструкция <row>. Если сумма значений **width** всех полей внутри <row> превышает 100, то возникнет ошибка при обработке сценария. Если сумма значений **width** всех полей внутри <row> превышает 100, но **width** указан не для всех полей, то возникнет ошибка при обработке сценария. Если значение не указано, то считается, что такое поле занимает все свободное место. Если таких полей несколько, то они делят все свободное место поровну. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper».	Нет

Атрибут	Описание	Обяз.
rowId	Идентификатор строки, в которой будет расположено поле. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	Нет
force-external-range	Позволяет принудительно устанавливать ограничение на ввод суммы. Возможные значения: • **true** — возможно установить максимальное и минимальное значения суммы, используя атрибуты **begin**, **end**; • **false** — ограничение на ввод суммы устанавливается извне. Пример использования атрибута: `<numeric-field id="sums" title="Сумма платежа" default="20" format="5.2" force-external-range="true"> <verify type="sumpurchase"> <range begin="0" end="15000"/> </verify> </numeric-field>` В данном примере будут установлено ограничение на вводимую сумму от 0 до 15000, которое задано в атрибутах **begin**, **end**.	Нет

Для проверки вводимого значения используется элемент, описываемый тегом **<verify>.** Вводимое значение должно включаться в интервал от «начальное_значение» до «конечное_значение».

Структура элемента:

```
<verify>
      <range begin=<float>
             end=<float>/>
</verify>
```

Дочерние теги: `<range>`.

Элемент имеет один атрибут **`type`**, который может принимать следующие значения:

1. **`range`** — значение по умолчанию. Можно не указывать.

2. **`sumpurchase`** — игнорирует атрибуты, задающие диапазон допустимых значений в сценарии. Диапазон допустимых значений суммы будет формироваться на этапе ввода исходя из настроек сервиса: нижняя граница станет равной минимальной сумме платежа по сервису, верхняя граница — это величина, при которой запрашиваемая сумма не превысит (с учетом комиссии) максимальной суммы платежа. Например, установлена минимальная сумма 10, максимальная — 500, минимальная и максимальная комиссия — 10%. В качестве суммы платежа максимально возможно указать 454,55, с учетом комиссии платеж составит 500. Минимальная сумма платежа — останется 10. С учетом комиссии платеж составит 11.

Пример:

```
<verify type="sumpurchase"/>
```

Для элемента `<range>` атрибуты приведены в таблице 5.8.2.

Таблица 5.8.2 — Атрибуты элемента `<range>`

Атрибут	Описание	Обяз.
`begin`	Начальное значение для проверки вводимого значения. В качестве значения данного атрибута возможно указать минимальную сумму для ввода в том случае, если элемент **`<verify>`** имеет тип **`sumpurchase`**	**Да**
`end`	Конечное значение для проверки вводимого значения. В в качестве значения данного атрибута возможно указать максимальную сумму для ввода в том случае, если элемент **`<verify>`** имеет тип	**Да**

Атрибут	Описание	Обяз.
	sumpurchase	

Пример:

```
<numeric-field id="sum" title="Сумма" format="7.2">
      <verify>
          <range begin="0" end="9999999.99"/>
      </verify>
</numeric-field>
```

В данное поле вводится переменная **sum** формата 7 цифр в целой части и 2 в дробной, принимаемые значения от 0 до 9999999.99.

Кроме того, в блок `<verify>` можно включать правила, осуществляющие проверку на соответствие регулярным выражениям.

Пример:

```
<verify>
      <rules>
          <rule regex="^086\d{7}$"/>
          <rule regex="^089\d{7}$"/>
          <block regex="^9\d{9}$"/>
      </rules>
</verify>
```

Пример:

```
<numeric-field id="id6" title="Сумма к оплате" format="6.2">
      <!--Подсказка, отображается на экране-->
      <help>
          <![CDATA[<html>Введите сумму к оплате<br>]]>
      </help>
      <verify>
          <range begin="10" end="999999"/>
      </verify>
</numeric-field>
```

Экран, соответствующий данному примеру, приведен на рисунке 5.8.2.

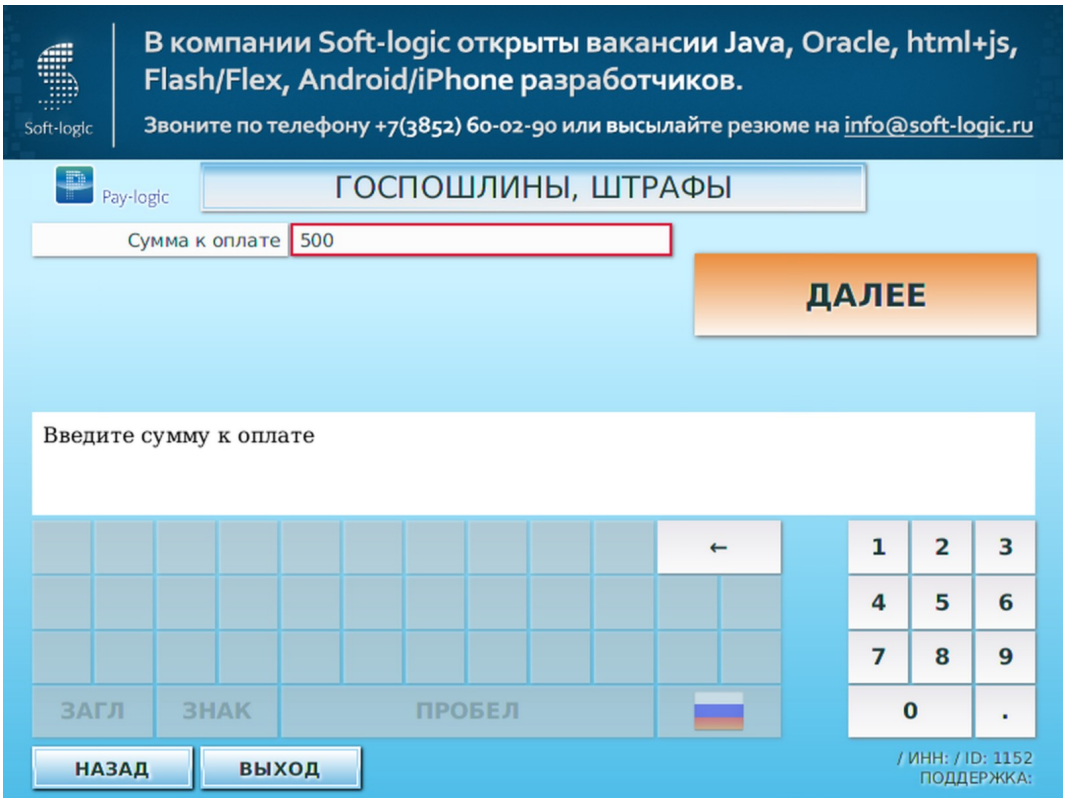


Рисунок 5.8.2 — Экран с элементом `<numeric-field>`

5.9 ФЛАГ/ПЕРЕКЛЮЧАТЕЛЬ (<CHECKBOX-FIELD>)

Элемент, описываемый тегом `<checkbox-field>`, позволяет управлять параметром с двумя состояниями: включено/выключено. Например, на экране может отображаться параметр «Согласен с условиями» с вариантами ответов «Да»/«Нет». Поддерживается обеими версиями ТПО. Структура элемента:

```
<checkbox-field id=<tId>
                group-id=<nonNegativeInteger>
                title=<titleType>
                title-id=<resourceIdType>
                message=<messageType>
                message-id=<resourceIdType>
                on-top=<tId>
                flags=<flags_specification>
                validate=[true|false]
                text=<notEmptyString40>
                message-off=<notEmptyString40>
                message-off-id=<resourceIdType>
                required=[true|false]
                width=<nonNegativeInteger>
                rowId=<nonNegativeInteger>>
      <help> ... </help>
      <help-off> ... </help-off>
</checkbox-field>
```

Дочерние элементы: `<help>` (раздел 5.13), `<help-off>` — аналогичен обычному `<help>`, но используется для задания текстовки подсказки для случая, когда флаг `<checkbox-field>` не установлен.

Атрибуты описаны в таблице 5.9.1.

Таблица 5.9.1 — Атрибуты элемента флаг/переключатель `<checkbox-field>`

Атрибут	Описание	Обяз.
`id`	Ключ атрибута платежа	**Да**
`group-id`	Значение, по которому происходит группировка полей на экране, то есть поля, для которых указано одно и то же значение **`group-id`**, визуально будут объединены на экране в один блок. Поддерживается некоторыми интерфейсами	Нет
`title`	Наименование атрибута платежа, в кабинете отображается в окне информации о платеже в блоке «Общая информация», также может добавляться в чек	Нет
`title-id`	Содержит идентификатор текстовки, которая будет подгружена в качестве значения **`title`**	**Да**
`message`	Заголовок экрана ввода	Нет
`message-id`	Идентификатор текстовки, которая будет использована в качестве сообщения заголовка экрана ввода	Нет
`on-top`	Позволяет редактировать значение поля ввода во всплывающем окне	Нет
`flags`	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет
`validate`	Определяет, когда валидно значение параметра. Возможные значения: • **`true`** — валидно при включенном состоянии; • **`false`** — всегда валидно. Значение по умолчанию	Нет
`text`	Текст самого переключателя. Может содержать илентификатор текстовки	Нет

Атрибут	Описание	Обяз.
message-off	Используется для указания заголовка экрана ввода, когда значение флага/переключателя не выбрано	Нет
message-off-id	Идентификатор текстовки, которая будет использована в качестве сообщения, когда значение флага/переключателя не выбрано	Нет
required	Определяет, когда валидно значение параметра. Возможные значения: • **true** — валидно при включенном состоянии; • **false** — всегда валидно. Значение по умолчанию	Нет
width	Ширина данного поля в текущей строке в процентах, если используется конструкция <row>. Если сумма значений **width** всех полей внутри <row> превышает 100, то возникнет ошибка при обработке сценария. Если сумма значений **width** всех полей внутри <row> превышает 100, но **width** указан не для всех полей, то возникнет ошибка при обработке сценария. Если значение не указано, то считается, что такое поле занимает все свободное место. Если таких полей несколько, то они делят все свободное место поровну. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper».	Нет
rowId	Идентификатор строки, в которой будет расположено поле. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	Нет

Пример:

```
<checkbox-field id="idcheckbox"
                title="Согласен с условиями"
                text="Установите отметку в случае согласия"
                message="Согласен"
                message-off="Не согласен"
                required="true">
      <help>
          Внимательно ознакомьтесь с условиями оплаты
      </help>
      <help-off>
          Оплата невозможна без согласия с условиями
      </help-off>
</checkbox-field>
```

Экран, соответствующий данному примеру, приведен на рисунке 5.9.1.

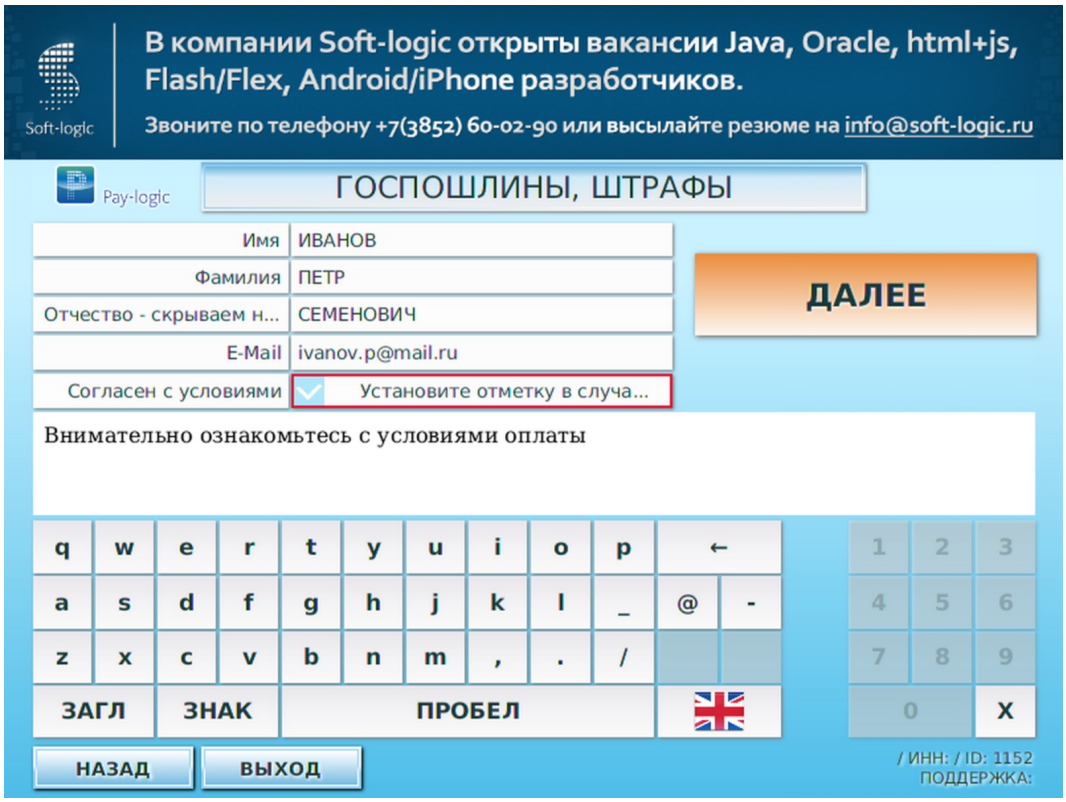


Рисунок 5.9.1 — Экран с элементом `<checkbox-field>`

5.10 СЕЛЕКТОР

5.10.1 ПРОСТОЙ СЕЛЕКТОР (<SELECTOR-FIELD>)

Селектор `<selector-field>` описывает список для выбора информации. На экране «SELECTOR» используется один элемент, на экране «GROUP» может быть использовано несколько элементов. Может отображать информацию для выбора в виде списка, кнопок с текстовой информацией, кнопок с текстовой информацией и подсказками к ним или в виде иконок с изображениями. Внешний вид одного из вариантов селектора приведен на рисунке 5.10.1.1.

Рисунок 5.10.1.1 — Внешний вид одного из вариантов селектора

Структура:

```
<selector-field id=<tId>
                group-id=<nonNegativeInteger>
                title=<titleType>
                title-id=<resourceIdType>
                message=<messageType>
                message-id=<resourceIdType>
                on-top=<tId>
                flags=<flags_specification>
                exist=[true|false]
                selected-index=<nonNegativeInteger>
                width=<nonNegativeInteger>
                rowId=<nonNegativeInteger>>
      <items type=[calendar|data|static]>
          <item .../>
          ...
      </items>
```

```
        ...
</selector-field>
```

Атрибуты `<selector-field>` описаны в таблице 5.10.1.1.

Таблица 5.10.1.1 — Атрибуты селектора `<selector-field>`

Атрибут	Описание	Обяз.
`id`	Ключ атрибута платежа	**Да**
`title`	Наименование атрибута платежа. В кабинете отображается в окне информации о платеже в блоке «Общая информация», также может добавляться в чек	**Да**
`title-id`	Содержит идентификатор локализованной текстовки, которая будет подгружена в качестве значения **`title`** Поддерживается только на ТПО 7	Нет
`exist`	Определяет показывать поле только, если в контексте есть соответствующая переменная или вне зависимости от этого. Возможные значения: • **`true`** — поле отображается только, если в контексте есть соответствующая переменная; • **`false`** — поле отображается вне зависимости от того, есть ли в контексте соответствующая переменная	Нет
`flags`	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет
`group-id`	Значение, по которому происходит группировка полей на экране, то есть поля, для которых указано одно и то же значение **`group-id`**, визуально будут объединены на экране в один блок. Поддерживается некоторыми интерфейсами	Нет
`message`	Заголовок экрана ввода	Нет

Атрибут	Описание	Обяз.
message-id	Идентификатор текстовки, которая будет использована в качестве сообщения заголовка экрана ввода	Нет
on-top	Позволяет редактировать значение поля ввода во всплывающем окне	Нет
rowId	Идентификатор строки, в которой будет расположено поле. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	Нет

selected-index	Индекс элемента, который будет выбран изначально:
	`selected-index="n"`
	где **n** — порядковый номер. Отсчет начинается с 0.
	Пример:
	`<screen type="selector/list" title="Экран выбора дня недели" id="select"><fields><selector-field id="type" title="Выбор дня недели" selected-index="2"><items type="static"><item value="1" title="Понедельник"/><item value="2" title="Вторник"/><item value="3" title="Среда"/><item value="4" title="Четверг"/><item value="5" title="Пятница"/><item value="6" title="Суббота"/><item value="7" title="Воскресенье"/></items></selector-field></fields>`
		Нет

width	Ширина данного поля в текущей строке в процентах,	Нет

Атрибут	Описание	Обяз.
	если используется конструкция <row>. Если сумма значений **width** всех полей внутри <row> превышает 100, то возникнет ошибка при обработке сценария. Если сумма значений **width** всех полей внутри <row> превышает 100, но **width** указан не для всех полей, то возникнет ошибка при обработке сценария. Если значение не указано, то считается, что такое поле занимает все свободное место. Если таких полей несколько, то они делят все свободное место поровну. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	

Дочерние элементы <selector-field>: <items>.

Для элемента **<items>** существует атрибут **type**, который может принимать следующие значения: calendar, data, static.

1. Тип **calendar** — выведет клиенту список автоматически сгенерированных дат, согласно заданным настройкам, фактически календарь с возможностью выбора даты. Для элемента <items> с типом **calendar** доступен дочерний элемент <init> с атрибутами, приведенными в таблице 5.10.1.2.

Таблица 5.10.1.2 — Атрибуты элемента <init> для <items type=**calendar**>

Атрибут	Описание	Обяз.
type	Задает тип хранилища. Может принимать значения Day, Month, Year (день, месяц, год) — для 7 версии ТПО. Обязательно указание типа с заглавной буквы. Для 5 версии ТПО не реализовано. При генерации элементов цикл будет происходить соответственно по	Да

Атрибут	Описание	Обяз.
	дням, месяцам или годам	
key-format	Задает формат значения даты. Формат даты задается в соответствии с SimpleDateFormat (день месяца — dd, месяц в году — MM, год — yyyy)	Да
title-format	Задает формат значения даты, выводимый клиенту	Да
offset	Задает смещение от текущей даты (может принимать как положительное, так и отрицательное значение)	Да
count	Задает количество выводимых записей в селекторе	Да

2. Тип **data** — данные возвращаются от провайдера. Для элемента `<items>` с типом `data` доступен атрибут **key=<string>** — непустая строка до 25 символов.

3. Тип **static** — данные прописываются в сценарии. Для тега `<items>` с типом `static` доступен дочерний элемент `<item>` с атрибутами, приведенными в таблице 5.10.1.3, обязательный атрибут **type=[static-extended|static].**

Таблица 5.10.1.3 — Атрибуты элемента `<item>` для `<items type=`**static**`>`

Атрибут	Описание	Обяз.
title	Используется для указания наименования атрибута платежа	Да
image	Указывается наименование файла логотипа с расширением .png с заданным размером «124x80» для 5 версии ТПО, «175x140» в 7 версии ТПО. В 7 версии ТПО изображения должны размещаться в каталоге ***<корень ТПО>/gui/skin/<наименование интерфейса>/img/***. В 5 версии ТПО — ***<корень ТПО>/img/<наименование интерфейса>/icon/service/***. И в 5, и в 7 версии ТПО реализована возможность	Нет

Атрибут	Описание	Обяз.
	создания вложенных папок. То есть в 7 версии при указании папки **image=«dir1/dir2/dir3/image1.png»** следует файл логотипа положить в каталог ***<корень ТПО>/gui/skin/<наименование интерфейса>/img/dir1/dir2/dir3/***, в 5 версии — ***<корень ТПО>/img/<наименование интерфейса>/icon/service/dir1/dir2/dir3/***	
value	Указывается значение атрибута платежа, добавляется в набор атрибутов при переходе на экран оплаты	Да
enabled	Указывается для активации или блокировки элемента списка. Возможные значения: • **true** — соответствует активному элементу (элемент доступен для выбора); • **false** — применяется для блокировки возможности выбора элемента пользователем. Для текстовых и графических селекторов работает одинаково. Если не указан, по умолчанию элемент активен	Нет
sum	Атрибут платежа, определенный как сумма по умолчанию, может быть указан дополнительно для отображения на экране фиксированной суммы оплаты по сервису	Нет

decor	Определяет внешний вид кнопки — позволяет использовать для нее определенную подложку или цвет. Текстовое поле. Виды интерфейсов, поддерживающих обработку значений данного параметра, уточняйте у сотрудников Soft-logic.
	Пример:
	`<selector-field id="id2" title="Тип оплаты">`
	`<items type="static">`
		Нет

Атрибут	Описание	Обяз.
	<item value="1" title="Погашение займа клиентом" decor="red"/> <item value="2" title="Платёж агента по одному договору" decor="green"/> <item value="3" title="Платёж агента по реестру договоров" /> </items> </selector-field>	

Пример статического селектора с атрибутом **<decor="red">** — на рисунке 5.10.1.2.

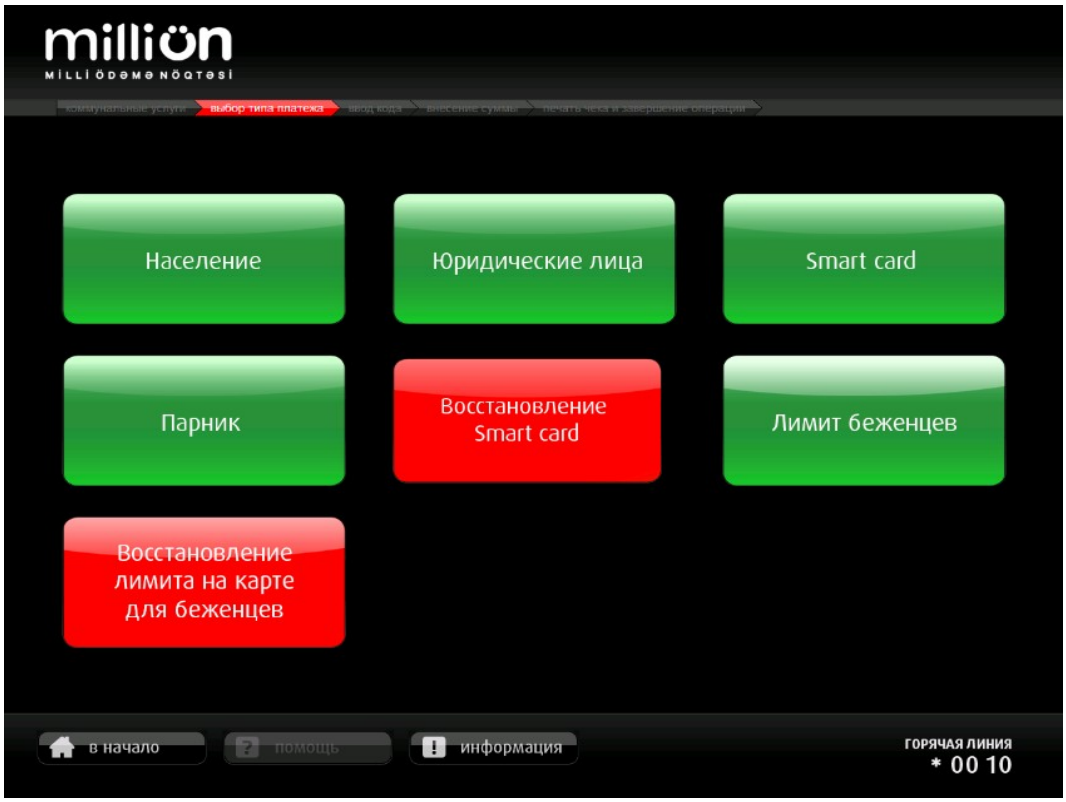


Рисунок 5.10.1.2 — Пример селектора с заданным атрибутом **decor**

Таким образом, в зависимости от оформления экрана выбора элемент `<selector-field>` имеет различную структуру.

ВАРИАНТ 1. ПРИ ИСПОЛЬЗОВАНИИ SCREEN (**DECOR="BUTTON"**, **DECOR="LIST"**)

При использовании значений **button**, **list** атрибута **decor** элемента <screen> при явном указании элементов <item> структура элемента следующая:

```
<selector-field id=<tId>
                title=<titleType>>
      <items type="static">
          <item title=<selectorTitleType>
                value=<notEmptyString>
                sum=<floatSum>/>
          <item title=<selectorTitleType>
                value=<notEmptyString>
                sum=<floatSum>/>
          ...
      </items>
</selector-field>
```

Причем, в случае **list** используется неограниченное число элементов <item>, в случае **button** используется не более 12 элементов <item>.

ВАРИАНТ 2. ПРИ ИСПОЛЬЗОВАНИИ SCREEN (**DECOR="IMAGE"**)

При использовании значения **image** атрибута **decor** элемента <screen> при явном указании элементов <item> структура элемента следующая:

```
<selector-field id=<tId>
                title=<titleType>>
      <items type="static">
          <item image=<fileNameType>
                value=<notEmptyString>
                sum=<floatSum>/>
          <item image=<fileNameType>
                value=<notEmptyString>
                sum=<floatSum>/>
          ...
      </items>
</selector-field>
```

Название изображения задается в виде название_файла.расширение_файла, например, *beeline.png*. В 7 версии ТПО изображения должны размещаться в каталоге ***<корень ТПО>/gui/skin/<наименование интерфейса>/img/***. В 5 версии ТПО — ***<корень ТПО>/img/<наименование интерфейса>/icon/service/***. И в 5, и в 7 версии ТПО

реализована возможность создания вложенных папок. То есть в 7 версии при указании папки **image=«dir1/dir2/dir3/image1.png»** следует файл логотипа положить в каталог ***<корень ТПО>/gui/skin/<наименование интерфейса>/img/dir1/dir2/dir3/***, в 5 версии — ***<корень ТПО>/img/<наименование интерфейса>/icon/service/dir1/dir2/dir3/***.

ВАРИАНТ 3. ПРИ ИСПОЛЬЗОВАНИИ SCREEN (**DECOR="VARIANT"**)

При использовании значения **variant** атрибута **decor** элемента `<screen>`, не более 4 элементов `<item>` при явном указании элементов `item` структура элемента следующая:

```
<selector-field id=<tId>
                title=<titleType>>
      <items type="static">
          <item title=<selectorTitleType>
                value=<notEmptyString>
                sum=<floatSum>>
                <help><![CDATA[Текст подсказки]]></help>
          </item>
          <item title=<selectorTitleType>
                value=<notEmptyString>
                sum=<floatSum>>
                <help><![CDATA[<html>Текст подсказки]]></help>
          </item>
          ...
      </items>
</selector-field>
```

Значение атрибута **sum** передается как сумма платежа.

Атрибуты **value** и **sum** являются необязательными, но, как правило, всегда присутствует один из них.

ВАРИАНТ 4. ПРИ ИСПОЛЬЗОВАНИИ **ONLINE-REQUEST**

При использовании онлайн-запроса `<online-request>` на предшествующем экране при неявном указании элементов `<item>` структура элемента следующая:

```
<selector-field id=<tId>
                title=<titleType>>
      <items type=[data|nested]
```

```
                key=<keyType>/>
</selector-field>
```

Список элементов <items> динамический, возвращается из-онлайн запроса:

1. **type** — указывается тип элементов из <online-request>. Допустимые значения **data**, **nested**.

2. **key** — указывается возвращаемый результат: название переменной онлайн-запроса.

Пример:

```
<selector-field id="ev_account1" title="Тарифность">
   <items type="static">
     <item value="0" title="без показаний приборов учета"/>
     <item value="1" title="однотарифный учет"/>
     <item value="2" title="два суточных показания"/>
     <item value="3" title="три суточных показания"/>
   </items>
</selector-field>
```

```
<selector-field id="fine" title="Постановления">
   <items type="data" key="fines"/>
</selector-field>
```

ВАРИАНТ 5. ИСПОЛЬЗОВАНИЕ <SELECTOR-FIELD> ДЛЯ ВЫВОДА ДАТЫ

Для более удобного выбора даты можно использовать селектор с типом списка элементов <items> **calendar**. Выводит клиенту список автоматически сгенерированных дат, согласно указанным настройкам.

Список элементов <items> имеет тип **calendar**.

Пример:

```
<selector-field id="srok" title="Срок действия">
   <items type="calendar">
     <init type="Month" key-format="MM.yy" title-format="MM/yy"
           offset="0" count="5"/>
   </items>
</selector-field>
```

Данный тип селектора может быть актуален для услуг ЖКХ (выбор периода оплаты), банковских карт (срок действия карт). Внутри элемента <item> возможно использование действия <set> для добавления inputElement-ов в контекст.

Пример:

```
<selector-field id="selector" title="Title of selector-field"
                title-id="title-id.selector">
  <items type="static">
    <item title="Выбор 1" value="1" title-id="title-id.item1">
      <set key="type" value="10" key-title-id="title-id.var1"/>
    </item>
  </items>
</selector-field>
```

Пример:

```
<selector-field id="numType" title="Вид платежа">
        <items type="static">
           <item title="№ Вод.Удостоверения" value="22"/>
           <item title="№ Св-ва о Регистрации ТС" value="24"/>
        </items>
</selector-field>
```

Экран, соответствующий данному примеру приведен на рисунке 5.10.1.3.

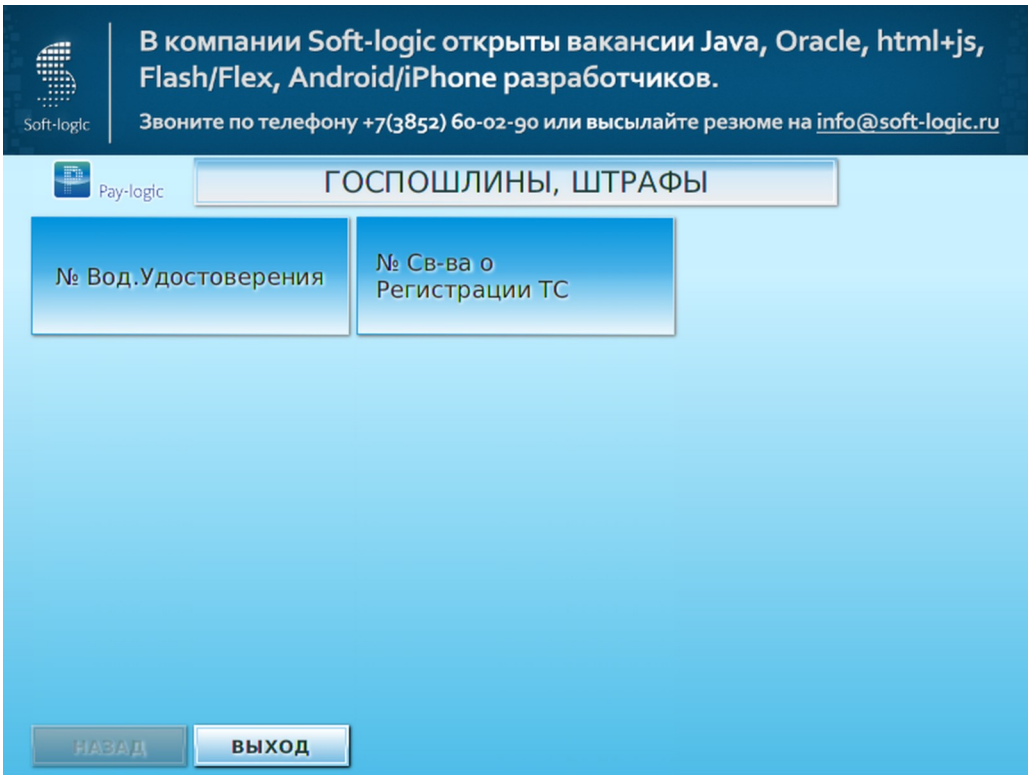


Рисунок 5.10.1.3 — Экран с элементом `<selector-field>`

5.10.2 ТАБЛИЧНЫЙ СЕЛЕКТОР (<TABLE-FIELD>)

Табличный селектор, описываемый тегом <table-field>, отображает информацию в виде таблицы. Выводит таблицу, позволяет выбирать строку данных таблицы, используется только на экранах «TABLE» и «TABLE/NAVI». Структура элемента:

```
<table-field id=<tId>
             group-id=<nonNegativeInteger>
             title=<titleType>
             title-id=<resourceIdType>
             message=<messageType>
             message-id=<resourceIdType>
             on-top=<tId>
             flags=<flags_specification>
             exist=[true|false]
             read-only=[true|false]>
             empty-text=<messageType>
             width=<nonNegativeInteger>
             rowId=<nonNegativeInteger>>
      <help>
          <string>
      </help>
      <columns>
          <column .../>
                ...
      </columns>
      <items>
          <item>
                <cell .../>
                …
          <item>
      </items>
</table-field>
```

Дочерние элементы: <help> (раздел 5.13), <columns>, <items>.

Атрибуты описаны в таблице 5.10.2.1.

Таблица 5.10.2.1 — Атрибуты табличного селектора

Атрибут	Описание	Обяз.
id	Ключ атрибута платежа	Да
group-id	Значение, по которому происходит группировка полей на экране, то есть поля, для которых указано одно и то же значение **group-id**, визуально будут объединены на экране в один блок. Поддерживается некоторыми интерфейсами	Нет
title	Наименование атрибута платежа, в кабинете отображается в окне информации о платеже в блоке «Общая информация», также может добавляться в чек	Нет
title-id	Содержит идентификатор текстовки, которая будет подгружена в качестве значения **title**	Нет
message	Заголовок экрана ввода	Нет
message-id	Идентификатор текстовки, которая будет использована в качестве сообщения заголовка экрана ввода	Нет
on-top	Позволяет редактировать значение поля ввода во всплывающем окне	Нет
flags	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет
exist	Определяет показывать поле только, если в контексте есть соответствующая переменная или вне зависимости от этого. Возможные значения: • **true** — поле отображается только, если в контексте есть соответствующая переменная; • **false** — поле отображается вне зависимости от того, есть ли в контексте соответствующая переменная	Нет

Атрибут	Описание	Обяз.
read-only	Флаг «только чтение», принимает одно из двух значений: • **true** — только чтение, поле для ввода неактивно; • **false** — поле для ввода активно, используется по умолчанию. В административной части системы электронных кошельков «SmartKeeper» поддерживается с версии 3.1	Нет
empty-text	Текст, который будет отображен в поле в качестве плейсхолдера (картинки-заглушки)	Нет
width	Ширина данного поля в текущей строке в процентах, если используется конструкция <row>. Если сумма значений **width** всех полей внутри <row> превышает 100, то возникнет ошибка при обработке сценария. Если сумма значений **width** всех полей внутри <row> превышает 100, но **width** указан не для всех полей, то возникнет ошибка при обработке сценария. Если значение не указано, то считается, что такое поле занимает все свободное место. Если таких полей несколько, то они делят все свободное место поровну. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper».	Нет
rowId	Идентификатор строки, в которой будет расположено поле. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	Нет

Элемент <columns> позволяет создать таблицу с необходимыми данными.

Дочерние элементы: <column> — задает название столбцов таблицы.

Атрибуты: отсутствуют.

Атрибуты элемента <column> приведены в таблице 5.10.2.2.

Таблица 5.10.2.2 — Атрибуты элемента **<column>**

Атрибут	Описание	Обяз.
key	Переменная либо из <item>, либо из VelocityContext с типом NestedData (объект заданный в виде «ключ-набор значений»)	Да
title	Отображаемое наименование переменной	Нет
title-id	Содержит идентификатор текстовки, которая будет подгружена в качестве значения **title**	Нет
width	Положительное дробное число, задающее ширину столбца таблицы. Сумма всех столбцов суммируется, принимается за единицу и ширина каждого столбца определяется в долях	Нет

is-total	Выводит сумму, общую для всех строк столбца. Может принимать значения **true** и **false.** Пример:
	`<columns>`
	`  <column key="#price" title="Цена:" width="0.3" is-total="true"/>`
	`</columns>`
	1. Если **true**, то на экране внизу таблицы будет выведена сумма по всем строкам колонки. В настоящее время поддерживается только интерфейсом Table (decor="Confirm").
		Нет

Дочерний элемент <column> — поле, описывающее столбец таблицы.

Структура элемента <column>:

```
<column key=<keyType>
        title=<titleType>
        title-id=<resourceIdType>
        width=<positiveFloat>>
</column>
```

Элемент <items> описывает строку таблицы и определяет тип таблицы с помощью атрибута **type**, который может принимать следующие значения:

1. **static** — статическая, данные таблицы описываются в элементах <items>, <columns>.

2. **data** — динамическая, данные берутся из context, используется с **key**. Атрибут **key** позволяет указать имя переменной из context в виде «ключ-значение». Обязательный атрибут. Работа со сложными структурами (NestedData) разобрана в разделе 7.

Структура элемента <items>:

```
<items>
       <item>
           <cell key=<keyType>
                 key-title=<titleType>
                 value=<notEmptyString>
                 value-title=<notEmptyString>/>
       ...
       </item>
       ...
</items>
```

При выборе строки таблицы:

1. В контекст распаковываются элементы, составляющие эту строку.

2. Переменная, ключ которой задан в атрибуте **id** элемента <table-field>, получает в качестве **value**, **value-title** значения элементов, составленные путем конкатенации значений из каждой ячейки <cell> выбранной строки через запятую. Пример рассмотрен в текущем разделе ниже.

Элемент <item (cell)> описывает значение каждой ячейки столбца таблицы.

Элемент <cell> позволяет получить значение каждой ячейки столбца таблицы, атрибуты приведены в таблице 5.10.2.3.

Таблица 5.10.2.3 — Атрибуты элемента <cell> для элемента <items> с типом **calendar**

Атрибут	Описание	Обяз.
key	Имя переменной столбца	Да
key-title	Отображаемое наименование переменной	Да
value	Значение ячейки указанного столбца	Да
value-title	Значение ячейки указанного столбца, выводимое на экране	Да

ВАРИАНТ 1. СТАТИЧЕСКАЯ ТАБЛИЦА СО ЗНАЧЕНИЯМИ ИЗ <ITEM> (<CELL>)

```
<scenario begin="1screen" chnum-name="id1">
 <!--Создание табличного экрана ввода данных-->
 <screen type="table" title="Введите данные" id="1screen">
  <fields>
   <!--Создание табличного поля ввода данных-->
   <table-field id="tf" title="Выберите маршрут" title-id=""
                read-only="false">
    <!--Описание заголовков граф таблицы-->
    <columns>
     <column key="#name" title="Маршрут:" title-id="" width ="0.5"/>
     <column key="#km" title="Расстояние:" width ="0.2"/>
     <column key="#price" title="Цена:" width ="0.3"/>
    </columns>
    <!--Описание ячеек таблицы-->
    <items type="static">
     <item>
      <cell key="#name" key-title="Маршрут" value="Барнаул-Алейск"
            value-title="Барнаул-Алейск"/>
      <cell key="#km" key-title="Расстояние" value="133"
            value-title="133 км"/>
      <cell key="#price" key-title="Цена" value="250"
            value-title="250 руб."/>
```

```
   </item>
   <item>
    <cell key="#name" key-title="Маршрут" value="Барнаул-Рубцовск"
          value-title="Барнаул-Рубцовск"/>
    <cell key="#km" key-title="Расстояние" value="275"
          value-title="275 км"/>
    <cell key="#price" key-title="Цена" value="500"
          value-title="500 руб."/>
   </item>
   <item>
    <cell key="#name" key-title="Маршрут" value="Барнаул-Заринск"
          value-title="Барнаул-Заринск"/>
    <cell key="#km" key-title="Расстояние" value="115"
          value-title="115 км"/>
    <cell key="#price" key-title="Цена" value="270"
          value-title="270 руб."/>
   </item>
  </items>
</table-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии "Далее"-->
   <action type="Next" title="Далее">
      <goto target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии "Назад"-->
   <action type="Prev" title="Назад">
    <goto target="previous"/>
   </action>
   <action type="Exit" title="Выход">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
</scenario>
```

Экран, соответствующий данному примеру, приведен на рисунке 5.10.2.1.

В этом примере переменная **`tf`** в качестве **`value`** получит значение «Барнаул-Рубцовск, 275, 500»; в качестве **`value-title`** — «Барнаул-Рубцовск, 275 км, 500 руб.»

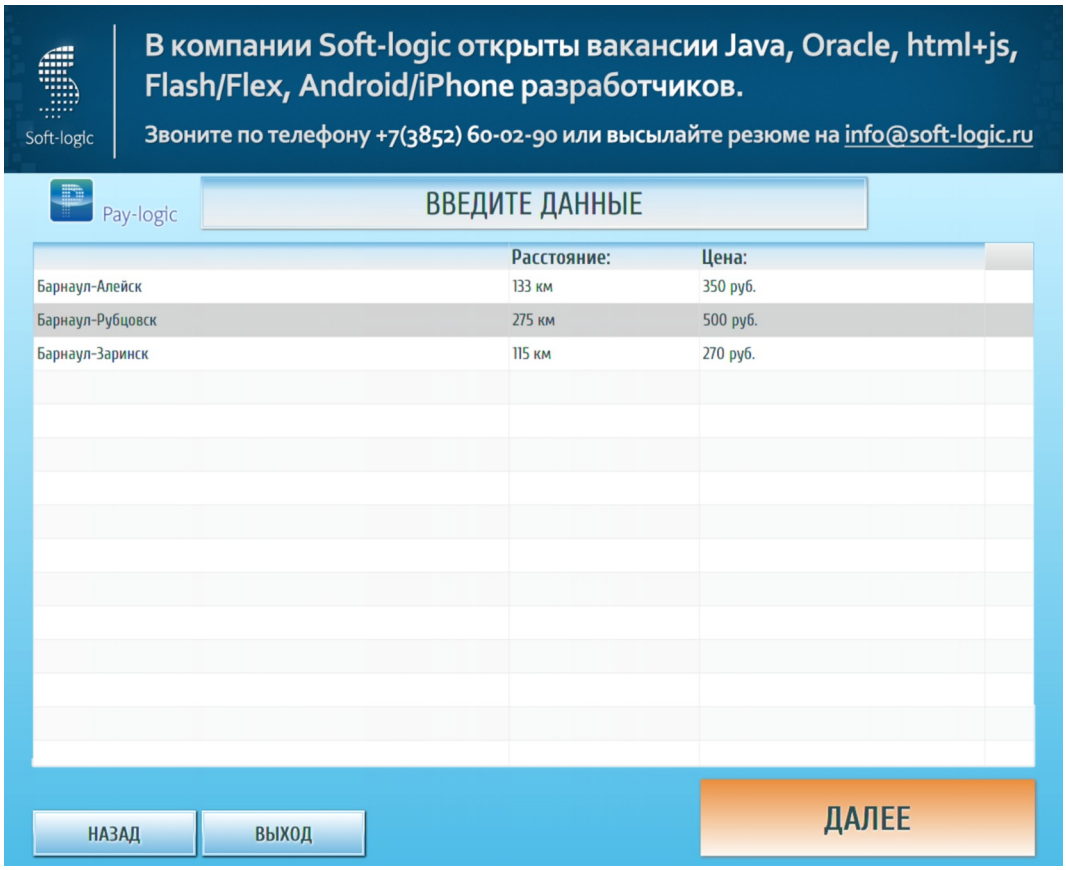


Рисунок 5.10.2.1 — Экран с элементом `<table-field>`

ВАРИАНТ 2. ДИНАМИЧЕСКАЯ ТАБЛИЦА СО ЗНАЧЕНИЯМИ ИЗ NESTED DATA

```
<!--Создание табличного экрана-->
<screen type="table" id="screen_table" title="Table экран">
   <fields>
     <!--Создание таблицы-->
     <table-field id="tf" title="Выберите маршрут">
       <items type="data"  key="#table_content"/>
       <!--Описание заголовков граф таблицы-->
       <columns>
         <column key="#name" title="Маршрут:" width ="0.5"/>
         <column key="#km" title="Расстояние:" width ="0.2"/>
         <column key="#price" title="Цена:" width ="0.3"/>
       </columns>
     </table-field>
```

```
  </fields>
  <actions>
    ...
  </actions>
</screen>
```

Данные формируются на предыдущем экране:

```
<!--Создание экрана формирования данных-->
<screen type="void" decor="simple" title="Формируем данные для table"
        id="void_data_to_table-field">
 <fields/>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии "Далее"-->
  <action type="Next" title="ДАЛЕЕ">
   <!--Объявление переменных-->
   <set key="#name" key-title ="Маршрут" value="Маршрут 1"
        value-title ="Маршрут 1"/>
   <set key="#km" value="25"/>
   <set key="#price" value="256"/>
   <!--Упаковка набора элементов-->
   <pack type="data" key="m1" elements="#name,#km,#price"/>
   <!--Объявление переменных-->
   <set key="#name" key-title ="Маршрут" value="Маршрут 2"
        value-title ="Маршрут 2"/>
   <set key="#km" value="21"/>
   <set key="#price" value="222"/>
   <!--Упаковка набора элементов-->
   <pack type="data" key="m2" elements="#name,#km,#price"/>
   <!--Объявление переменных-->
   <set key="#name" key-title ="Маршрут"  value="Маршрут 3"
        value-title ="Маршрут 3"/>
   <set key="#km" value="23"/>
   <set key="#price" value="344"/>
   <!--Упаковка набора элементов-->
   <pack type="data" key="m3" elements="#name,#km,#price"/>
   <!--Объявление переменных-->
   <set key="#name" key-title ="Маршрут"  value="Маршрут 4"
        value-title ="Маршрут 4"/>
   <set key="#km" value="43"/>
   <set key="#price" value="564"/>
   <!--Упаковка набора элементов-->
   <pack type="data" key="m4" elements="#name,#km,#price"/>
```

```
    <pack type="nested" key="#table_content" elements="m1,m2,m3,m4"/>
    <!--Удаление переменных из контекста-->
    <clear keys="#name,#km,#price,m1,m2,m3,m4"/>
    <goto target="screen_table"/>
   </action>
  </actions>
</screen>
```

ВАРИАНТ 3. НАВИГАЦИЯ

```
<!--Создание табличного экрана с навигацией-->
<screen type="table/navi" id="screen_table" title="Table экран">
  <fields>
    <!--Создание таблицы-->
    <table-field id="tf" title="Выберите маршрут" >
      <items type="data" key="#table_content"/>
      <!--Описание заголовков граф таблицы-->
      <columns>
        <column key="#name" title="Маршрут:" width ="0.5"/>
        <column key="#km" title="Расстояние:" width ="0.2"/>
        <column key="#price" title="Цена:" width ="0.3"/>
      </columns>
    </table-field>
  </fields>
  <!--Создание навигации на экране-->
  <navigation>
    <!--Создание первой кнопки навигации-->
    <action type="navi1" title="Выбор даты">
      <goto target="selector_screen2"/>
    </action>
    <!--Создание второй кнопки навигации-->
    <action type="navi2" title="Выбор маршрута"       >
      <goto target="group_screen"/>
    </action>
    <!--Создание третьей кнопки навигации-->
    <action type="navi3" title="Маршрут" current="true">
      <goto target="screen_table"/>
    </action>
  </navigation>
    <actions>
      ...
    </actions>
  </screen>
```

Экран, соответствующий данному примеру приведен в разделе 4.6.44 на рисунке 4.6.44.1. В элементе `<navigation>` описываются кнопки перехода к предыдущим экранам выбора даты и маршрута. Кнопка перехода на текущий экран неактивна и подсвечивается другим цветом.

5.11 ВЫБОР ДАТЫ (<DATE-FIELD>)

Поле выбора даты описывается элементом `<date-field>` и позволяет отображать календарь для выбора даты. Внешний вид приведен на рисунке 5.11.1. Используется только на групповом экране (раздел 4.6.18).

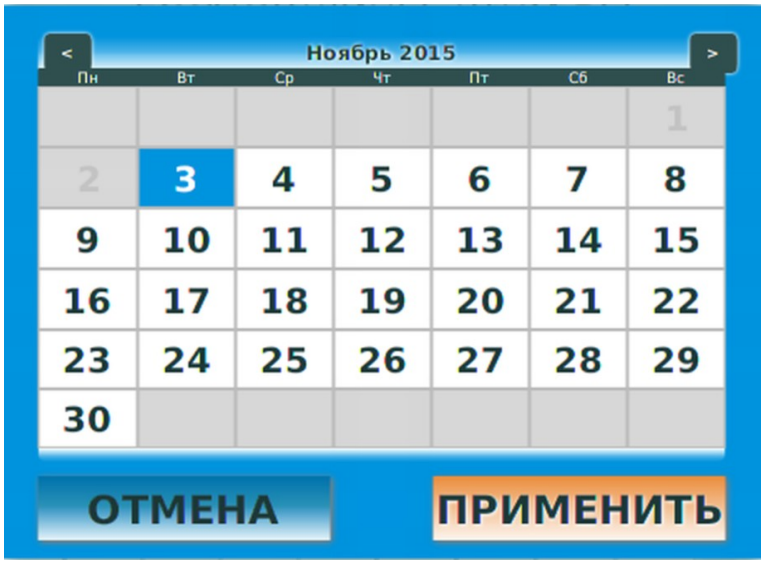


Рисунок 5.11.1 — Внешний вид элемента `<date-field>`

Структура элемента:

```
<date-field id=<tId>
            group-id=<nonNegativeInteger>
            title=<titleType>
            title-id=<resourceIdType>
            message=<messageType>
            message-id=<resourceIdType>
            on-top=<tId>
            flags=<flags_specification>
            exist=[true|false]
            from=<notEmptyString>
```

```
to=<notEmptyString>
default=<notEmptyString>
format=<notEmptyString>
format-title=<notEmptyString>
required=[true|false]
width=<nonNegativeInteger>
rowId=<nonNegativeInteger>/>
```

Дочерние элементы: `<help>` (раздел 5.13).

Атрибуты описаны в таблице 5.11.1.

Таблица 5.11.1 — Атрибуты `<date-field>`

Атрибут	Описание	Обяз.
id	Ключ атрибута платежа	Да
group-id	Значение, по которому происходит группировка полей на экране, то есть поля, для которых указано одно и то же значение **group-id**, визуально будут объединены на экране в один блок. Поддерживается некоторыми интерфейсами	Нет
title	Наименование атрибута платежа, в кабинете отображается в окне информации о платеже в блоке «Общая информация», также может добавляться в чек	Нет
title-id	Содержит идентификатор текстовки, которая будет подгружена в качестве значения **title**	Да
message	Заголовок экрана ввода	Нет
message-id	Идентификатор текстовки, которая будет использована в качестве сообщения заголовка экрана ввода	Нет
on-top	Позволяет редактировать значение поля ввода во всплывающем окне	Нет
flags	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться	Нет

Атрибут	Описание	Обяз.
	как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	
exist	Определяет показывать поле только, если в контексте есть соответствующая переменная или вне зависимости от этого. Возможные значения: • **true** — поле отображается только, если в контексте есть соответствующая переменная; • **false** — поле отображается вне зависимости от того, есть ли в контексте соответствующая переменная	Нет
from	Указывается дата в том же формате, что указан в format или число — количество предшествующих дней до текущей даты, в данном случае будет являться нижней границей заданного периода. Например, фиксированное значение «03.03.2014», то есть будет доступен выбор даты с 03.03.2014; динамическое значение — «0d», то есть будет доступен выбор даты с текущего дня. Возможно установить дату начала, используя значение атрибута. Например, если указать: `from="${suspendBeginDate}"` то дата начала заморозки контракта будет равна значению атрибута **suspendBeginDate**. Реализовано для базового группового экрана.	Нет
to	Указывается дата в том же формате, что указана в format или число: количество последующих дней от текущей даты, в данном случае будет являться верхней границей заданного периода. Например, фиксированное значение «05.04.2014», то есть будет доступен выбор даты до 05.04.2014; динамическое значение — «1m», то есть будет доступен выбор даты	Нет

Атрибут	Описание	Обяз.
	в течение 1 месяца от текущей. Возможно установить дату окончания, используя значение атрибута. Например, если указать: `to="${canSuspendDays}"` то дата окончания заморозки контракта будет соответствовать значению атрибута **`canSuspendDays`**. Реализовано для базового группового экрана.	
`default`	Указывается дата по умолчанию для отображения в поле ввода или число: количество последующих дней от текущей даты. Если не указан, то считается, что необходимо использовать дату текущего дня. При этом, если дата текущего дня не попадает в заданный параметрами **`from`**, **`to`** период, то возникнет ошибка при обработке сценария. Для таких случаев необходимо указывать значение несмотря на то, что атрибут не является обязательным. Возможно указывать как фиксированные, так и динамические значения. Например, фиксированное значение «03.03.2014», то есть по умолчанию будет выбрана дата 03.03.2014; динамическое значение — «0», то есть по умолчанию будет выбран текущий день	Нет
`format`	Указывается дата в указанном формате. DD — день, MM — месяц, yyyy — год. Возможный формат соответствует формату `SimpleDateFormat` (для 5 ТПО — https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html, для 7 ТПО — https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html)	Да
`format-title`	В указанном формате отображается пользователю. До	Нет

Атрибут	Описание	Обяз.
	версии ТПО 7.7 **formattitle**	
required	Используется для указания того, обязательным ли является поле <date-field>. Возможные значения: • **true** — является обязательным; • **false** — не является обязательным	Нет
width	Ширина данного поля в текущей строке в процентах, если используется конструкция <row>. Если сумма значений **width** всех полей внутри <row> превышает 100, то возникнет ошибка при обработке сценария. Если сумма значений **width** всех полей внутри <row> превышает 100, но **width** указан не для всех полей, то возникнет ошибка при обработке сценария. Если значение не указано, то считается, что такое поле занимает все свободное место. Если таких полей несколько, то они делят все свободное место поровну. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	Нет
rowId	Идентификатор строки, в которой будет расположено поле. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	Нет

Пример:

```
<!--Создание экрана ввода данных с несколькими полями-->
<screen type="group" title="Введите данные" id="input1">
      <fields>
          <!--Создание поля ввода-->
          <text-field id="id1" ... > ... </text-field>
          <!--Создание поля ввода-->
```

```
            <numeric-field id="sum1" ... > ... </numeric-field>
            <!--Создание поля выбора даты-->
            <date-field id="id2" title="Дата"
                        format="dd.MM.yyyy"
                        format-title="yyyy-MM-dd"
                        from="01.03.2014" to="05.04.2014"
                        default="03.03.2014"/>
      </fields>
      <actions> ... </actions>
</screen>
```

Экран, соответствующий данному экрану, приведен на рисунке 5.11.2.

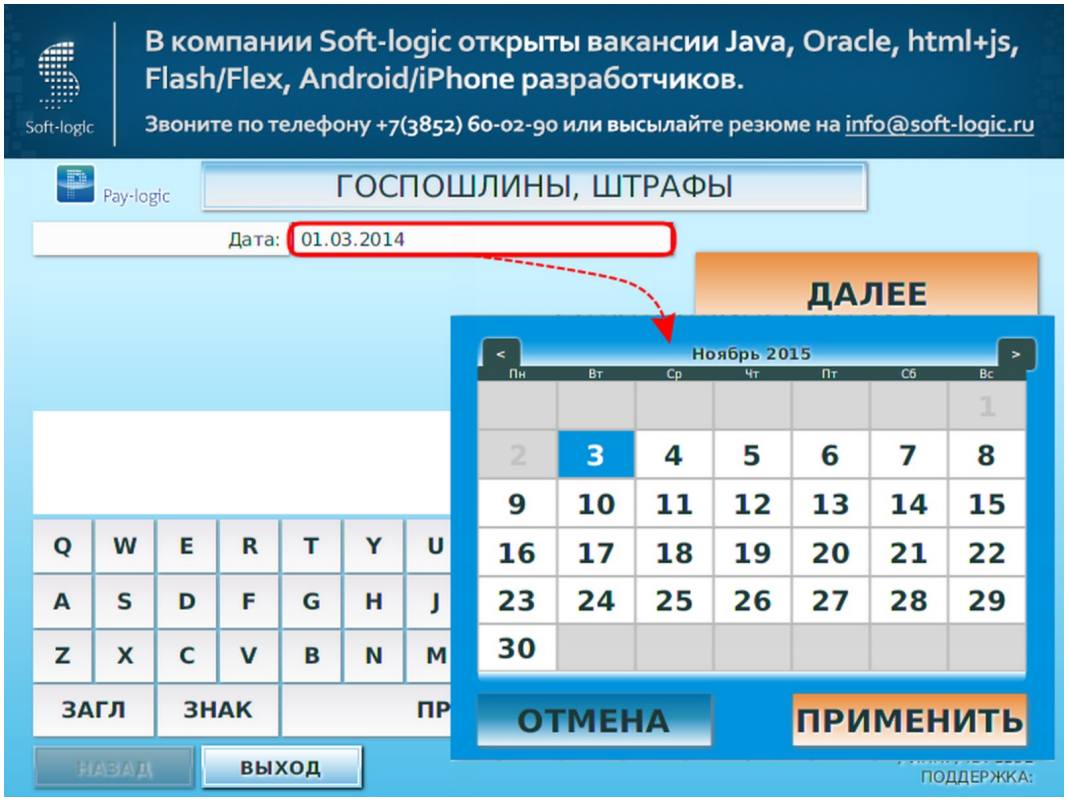


Рисунок 5.11.2 — Экран с элементом `<date-field>` с фиксированными значениями

Пример:

```
<!--Создание экрана ввода данных с несколькими полями-->
<screen type="group" title="Введите данные" id="input1">
 <fields>
   <!--Создание поля ввода-->
   <text-field id="id1" ... > ... </text-field>
   <!--Создание поля ввода-->
   <numeric-field id="sum1" ... > ... </numeric-field>
   <!--Создание поля выбора даты-->
   <date-field id="id2" title="Дата"
      format="dd.MM.yyyy"
      format-title="yyyy-MM-dd"
      from="0d" to="1m"       default="0"/>
 </fields>
 <actions> ... </actions>
</screen>
```

Экран, соответствующий данному экрану, приведен на рисунке 5.11.3.

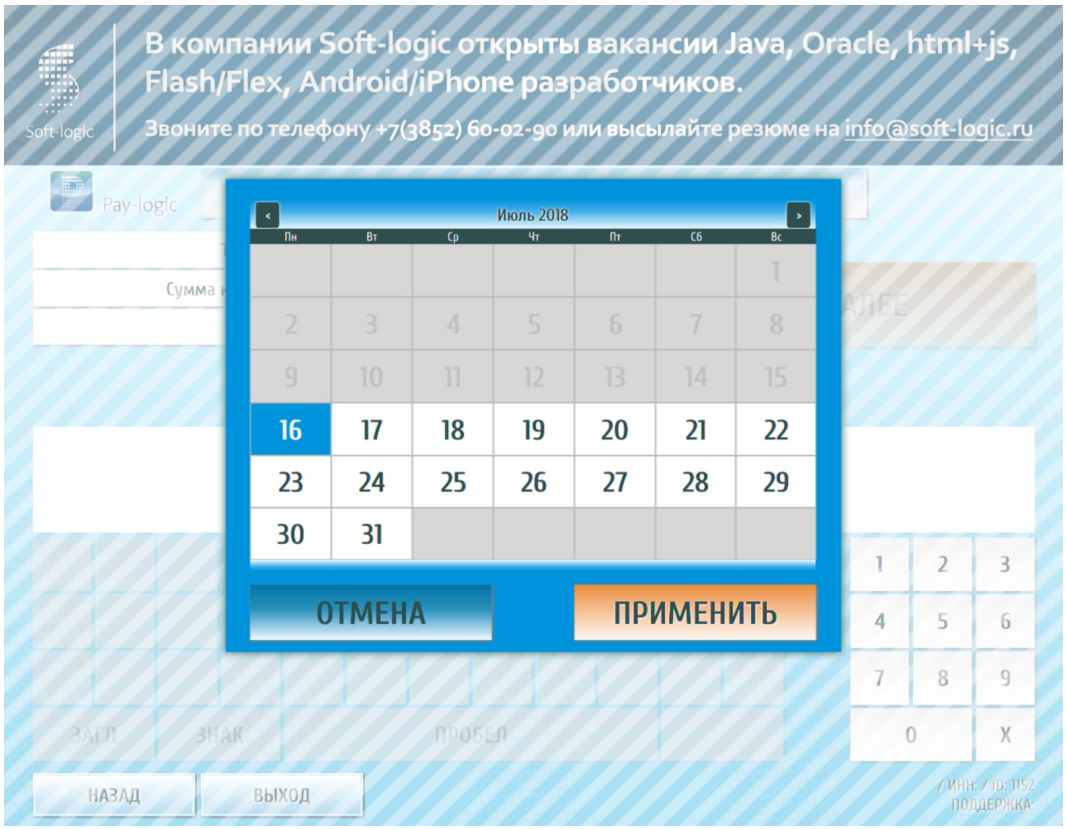


Рисунок 5.11.3 — Экран с элементом `<date-field>` с динамическими значениями

Используя `<set>` в сценарии возможно получить и использовать текущую дату, Подробнее в разделе 6.13.

5.12 АВТОЗАПОЛНЕНИЕ (<AUTOCOMPLETE-FIELD>)

Автозаполнение, описывается элементом `<autocomplete-field>`, отображает поле ввода, которое позволяет по введенной части слова выбирать значение из выпадающего списка. Список возможных значений определяется источником (элемент `<data-source>`). Внешний вид элемента приведен на рисунке 5.12.1. Используется только на групповом экране (раздел 4.6.18).

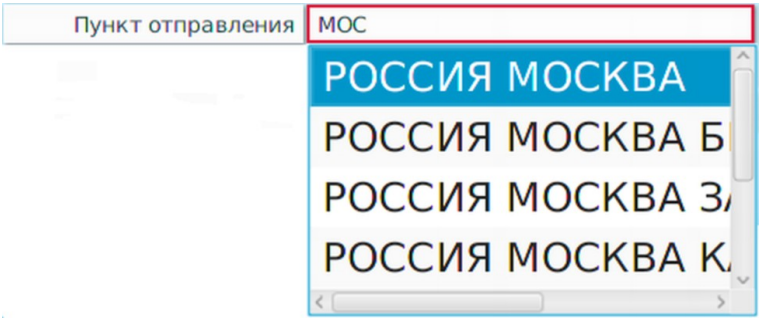


Рисунок 5.12.1 — Внешний вид элемента `<autocomplete-field>`

Структура элемента:

```
<autocomplete-field id=<tId>
                    group-id=<nonNegativeInteger>
                    title=<titleType>
                    title-id=<resourceIdType>
                    message=<messageType>
                    message-id=<resourceIdType>
                    flags=<flags_specification>
                    exist=[true|false]
                    min-chars=<positiveInteger>
                    max-items=<positiveInteger>
                    optional=[true|false]
                    keyboard=<keyboard_specification>
                    xbutton=<notEmptyString1>
                    letter-case=[UpperCase|LetterCase]
```

```
                    width=<nonNegativeInteger>
                    rowId=<nonNegativeInteger>
                    allow-custom=[true|false]>
      <data-source type=[plain]
                   partial=[none]
                   url=<notEmptyString>
                   default=<notEmptyString>
                   order=[rate|name]/>
      <help>
          <string>
      </help>
</autocomplete-field>
```

Дочерние элементы: `<data-source>`, `<help>` (раздел 5.13).

Атрибуты описаны в таблице 5.12.1.

Таблица 5.12.1 — Атрибуты элемента `<autocomplete-field>`

Атрибут	Описание	Обяз.
id	Ключ атрибута платежа	**Да**
group-id	Значение, по которому происходит группировка полей на экране, то есть поля, для которых указано одно и то же значение **group-id**, визуально будут объединены на экране в один блок. Поддерживается некоторыми интерфейсами	Нет
title	Наименование атрибута платежа, в кабинете отображается в окне информации о платеже в блоке «Общая информация», также может добавляться в чек	Нет
title-id	Содержит идентификатор текстовки, которая будет подгружена в качестве значения **title**	**Да**
message	Заголовок экрана ввода	Нет
message-id	Идентификатор текстовки, которая будет использована в качестве сообщения заголовка экрана ввода	Нет

flags	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет
exist	Определяет показывать поле только, если в контексте есть соответствующая переменная или вне зависимости от этого. Возможные значения: • **true** — поле отображается только, если в контексте есть соответствующая переменная; • **false** — поле отображается вне зависимости от того, есть ли в контексте соответствующая переменная	Нет
on-top	Позволяет редактировать значение поля ввода во всплывающем окне	Нет
flags	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет
min-chars	Минимальное количество символов, начиная с которого ищется список значений из файла источника (начинает работать поиск значений)	**Да**
max-items	Максимальное количество найденных значений из файла источника, удовлетворяющее введенной части слова в поле ввода, больше которого считать фразу недостаточной для осуществления поиска и не отображать найденные значения	**Да**
optional	Определяет является ли заполнение поля обязательным. Возможные значения: • **true** — поле обязательно для заполнения; • **false** — поле может оставаться пустым	Нет
keyboard	Указывается тип клавиатуры, отображаемой на экране терминала. Типы клавиатур описаны в разделе 5.6	Да

xbutton	Указывается один символ из символьной или буквенной раскладки клавиатуры, который будет отображаться вместо функциональной кнопки «Стереть» (или сброс введенного значения) для цифровой раскладки. Данный атрибут используется только с цифровой раскладкой. При указании клавиатуры в 5 версии ТПО важен регистр (digital — неверно)	Нет
letter-case	Предоставляет возможность установить первую букву сообщения заглавной. Возможные значения: • **UpperCase** — заглавная буква; • **LowerCase** — строчная буква	Нет
width	Ширина данного поля в текущей строке в процентах, если используется конструкция <row>. Если сумма значений **width** всех полей внутри <row> превышает 100, то возникнет ошибка при обработке сценария. Если сумма значений **width** всех полей внутри <row> превышает 100, но **width** указан не для всех полей, то возникнет ошибка при обработке сценария. Если значение не указано, то считается, что такое поле занимает все свободное место. Если таких полей несколько, то они делят все свободное место поровну. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	Нет
rowId	Идентификатор строки, в которой будет расположено поле. Поддержка атрибута зависит от интерфейса. В текущей версии реализовано только для «Платформы электронных кошельков Smartkeeper»	Нет
allow-custom	Параметр определяет возможность ввода отсутствующих в справочнике значений. Доступен в	Нет

ТПО 7, в версии 5 недоступен.

Возможные значения:

- **`true`** — разрешает принимать значения, отсутствующие в справочнике. Если пользователь вводит значение, ввел его не полностью и ничего не выбрал, либо ввел его таким, которое приводит к пустому списку выбора, поле принимает введенное значение в качестве значения;
- **`false`** — запрещает принимать значения, отсутствующие в справочнике. Если значение из предлагаемых не выбрано, поле остается пустым. Значение по умолчанию.

В совокупности с параметром **`optional`** реализуется следующая логика работы:

optional	allow-custom	Введено значение		
		пустое	не из списка	из списка
`false`	`false`	кнопка «Далее» неактивная	кнопка «Далее» неактивная	кнопка «Далее» активная
`false`	`true`	кнопка «Далее» неактивная	кнопка «Далее» активная	кнопка «Далее» активная
`true`	`false`	кнопка «Далее» активная	кнопка «Далее» неактивная	кнопка «Далее» активная
`true`	`true`	кнопка «Далее»	кнопка «Далее»	кнопка «Далее»

optional	**allow-custom**	**Введено значение**			
		активная	активная	активная	

Пример:

```
<autocomplete-field id="departure-point"
    title="Пункт отправления"
    min-chars="3" max-items="7"
    keyboard="any:[ru,en,symb]:lower:true"
    optional="false" allow-custom="true">
        <data-source type="plain"
            url="stations.dat"
            partial="none" />
            <help>
                <![CDATA[<html>Введите пункт
                отправления]]>
            </help>
</autocomplete-field>
```

Пример:

```
<autocomplete-field id="point" title="Пункт отправления" min-chars="3"
                    max-items="7" optional="false"
                    keyboard="any:[ru,en,dgtl,symb]:lower:true">
   <data-source type="plain" url="stations.dat" partial="none"
                order="name"/>
     <help>
       <![CDATA[<html>Введите пункт отправления]]>
     </help>
</autocomplete-field>
```

Экран, соответствующий данному примеру, приведен на рисунке 5.12.2.

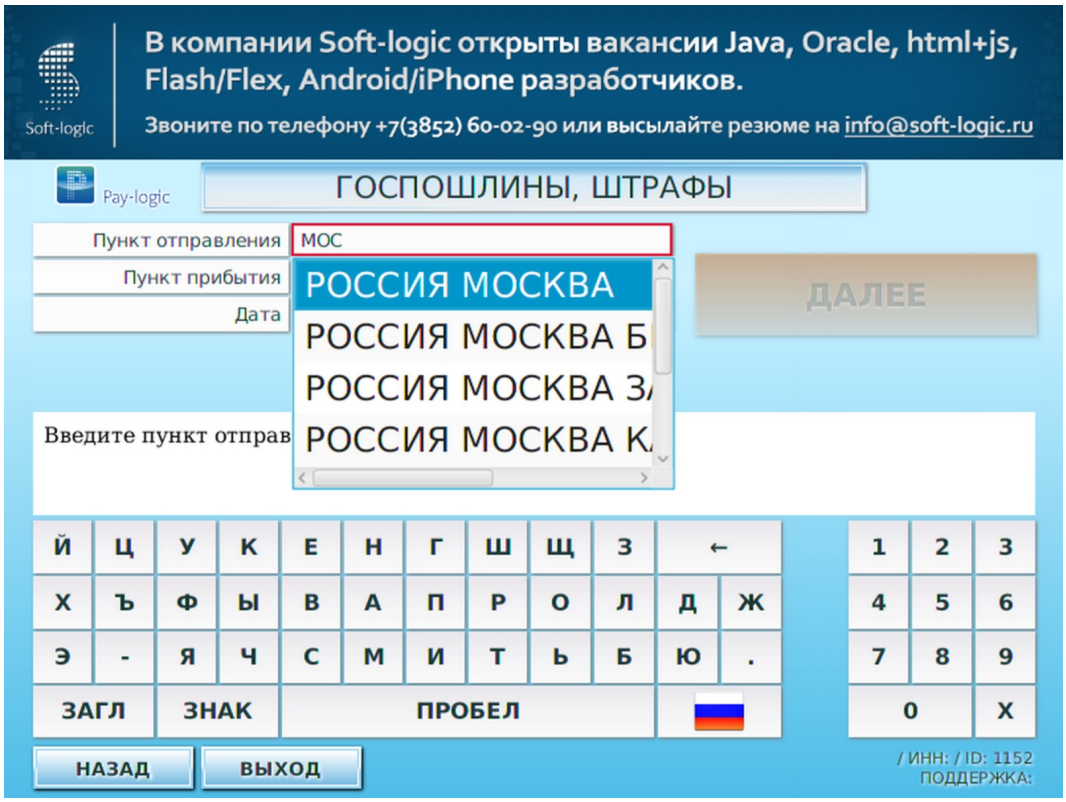


Рисунок 5.12.2 — Экран с элементом `<autocomplete-field>`

Описание источника данных содержит элемент, задаваемый элементом `<data-source>`.

Атрибуты описаны в таблице 5.12.2.

Таблица 5.12.2 — Атрибуты элемента `<data-source>`

Атрибут	Описание	Обяз.
`type`	Указывается тип источника данных. Определены значения: **`plain`** — источником данных служит файл	Да
`partial`	Указывается тип алгоритма поиска, определены значения: **`none`**. По умолчанию ищутся строки в файле-источнике по введенной части слова в поле	Нет

Атрибут	Описание	Обяз.
	ввода	
url	Указывается путь до источника данных. Поддерживается в ТПО 5, 7: `l="stations.dat"` Источник данных должен быть расположен: 1. Для ТПО 7 — по пути *<корень ТПО>/res/module/input/ds/<имя файла>.dat.* 2. Для ТПО 5 — по пути *<корень ТПО>/resources/autocomplete/<имя файла>.dat*	Да
default	Используется для указания значения, выводимого в поле ввода по умолчанию, можно редактировать	Нет
order	Доступен в ТПО 7. Определяет вариант сортировки. Возможны следующие значения: • **name** — сортировка по отображаемому названию. Значение по умолчанию; • **rate** — сортировка по рейтингу	Нет

Пример:

```
<autocomplete-field id="point" title="Пункт отправления" min-chars="3"
                    max-items="7" optional="false"
                    keyboard="any:[ru,en,dgtl,symb]:lower:true">
   <data-source type="plain" url="1152/stations.dat" partial="none"
                order="rate"/>
   <help> Введите пункт отправления </help>
</autocomplete-field>
```

Пример содержимого файла источника данных stations.dat:

```
2000000[18]:РОССИЯ МОСКВА
2000001[16]:РОССИЯ МОСКВА КУРСКАЯ
2000002[14]:РОССИЯ МОСКВА ЯРОСЛАВСКАЯ
2000003[11]:РОССИЯ МОСКВА КАЗАНСКАЯ
2000004[15]:РОССИЯ МИИТОВСКАЯ
```

```
2000005[13]:РОССИЯ МОСКВА ПАВЕЛЕЦКАЯ
2000006[12]:РОССИЯ МОСКВА БЕЛОРУССКАЯ
2000007[1]:РОССИЯ МОСКВА КИЕВСКАЯ
2000008[2]:РОССИЯ МОСКВА РИЖСКАЯ
2000009[3]:РОССИЯ МОСКВА САВЕЛОВСКАЯ
2000010[4]:РОССИЯ МОСКВА ЗАПАСНОЙ КОД
2000011[5]:РОССИЯ ЭЛЕКТРОГОРСК
2000012[6]:РОССИЯ ВАЛЕНТИНОВКА
2000013[7]:РОССИЯ ТРУДОВАЯ
2000014[8]:РОССИЯ ОСТ.ПУНКТ 37 КМ
2000015[9]:РОССИЯ МЫТИЩИ
2000016[10]:РОССИЯ ЦИОЛКОВСКАЯ
2000017[17]:РОССИЯ БОРДАВИЧИ
```

Структура файла: значение, указанное в квадратных скобках «[]», представляет собой вес значения. Доступно в ТПО 7. Большие значения в списке вариантов отображаются первыми среди вариантов при установленном **`order="rate"`** в `<data-source/>`. Значение до «[]» является значением создаваемого атрибута платежа, вторая строка (после символа «[]» — заголовок для значения атрибута платежа. Если нет символа «:», то значение строки станет и значением, и заголовком. Все значения после второго «:» распознаются, как альтернативное название. Количество альтернативных названий неограниченно, указываются через «:».

То есть, формат строки содержимого файла источника имеет следующий вид:

```
value[rate]:valueTitle:altName1,altName2, ... , altNameN
```

Значение элементов следующее:

1. **`value`** — значение элемента. Обязательное поле.
2. **`[rate]`** — рейтинг. По умолчанию **`rate = 0`**.
3. **`valueTitle`** — заголовок элемента. По умолчанию **`valueTitle = value`**.
4. **`altName1, altName2, ... , altNameN`** — альтернативные названия. По умолчанию отсутствуют.
5. Если используется сортировка **`order="name"`**, то **`[rate]`** из названия исключается и нигде не используется.

5.13 ПОДСКАЗКА (<HELP>)

Элемент <help> предназначен для вывода текста подсказки на экран. Для представления текста в форматированном виде допустимо использование html-тегов. Для этого используйте тег <html> внутри элемента <![CDATA[]]>.

```
<help>
   <![CDATA[<html>Текст подсказки]]>
</help>
```

Текст подсказки — используется язык гипертекстовой разметки:

```
<help>
   <![CDATA[<html>Укажите <b>НОМЕР ТЕЛЕФОНА</b> (10 цифр) для
            обратной связи.<br> В случае проблем с зачислением
            денежных средств мы с Вами свяжемся по указанному
            номеру.<br>]>
</help>
```

В 7 версии ТПО в элементе <help> возможно указания типа точки. Например, <help types="4"> — для электронного кошелька. Если не будет найдена подсказка для типа точки, то будет использована подсказка по умолчанию.

Возможные флаги:

1. **TERMINAL=0x01** — подсказка для терминала.
2. **RMA=0x02** — подсказка для РМА.
3. **KEEPER=0x04** — подсказка для кошелька.
4. **KEEPER-ANDROID=0x08** — подсказка для кошелька Android.
5. **KEEPER-IOS=0x10** — подсказка для iOS-клиентов.

6. **KEEPER-WEB=0x20** — подсказка для веб-клиентов.

7. **MOBILE-ANDROID=0x40** — подсказка для терминалов Android.

8. **MOBILE-IOS=0x80** — подсказка для терминалов iOS.

Пример:

```
<text-field>
      <help>
          Введите номер телефона
      </help>
      <help types="4">
          Введите номер телефона или выбирите его из вашей телефонной
          книги
      </help>
      <help types="RMA | TERMINAL">
          Введите номер телефона, после чего нажмите кнопку ДАЛЕЕ
      </help>
</text-field>
```

На экранах типа «DIGITAL» в качестве подсказки возможно использование изображений (изображения хранятся в ***<корень ТПО>/img/bluesphere2/helps***, размер изображения в пикселях 760*465):

```
<help>
      file://helps/217.2.png
</help>
```

Пример:

```
<!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 10. Название поля
ввода: "Номер телефона"-->
<text-field id="id2" title="Номер телефона"
            max-len="10" keyboard="Digital">
  <!--Подсказка, отображается на экране-->
  <help>
    <![CDATA[<html>Укажите <b>НОМЕР ТЕЛЕФОНА</b> (10 цифр) для
            обратной связи.<br>В случае проблем с зачислением денежных
            средств мы с Вами свяжемся по указанному номеру.<br>
      <terminal>
        Для перехода к оплате нажмите <b>"ДАЛЕЕ"</b>.
```

```
      </terminal>]]>
  </help>
  <validator type="regex">
    <rules> <rule regex="^\d{10}$"/> </rules>
  </validator>
  <!--Регулярное выражение для форматирования вводимого номера на
экране-->
  <formatter type="regex">
    <rules default="***-***-****"/>
  </formatter>
</text-field>
```

Кроме того, при выводе подсказок возможно использовать `<help>` с атрибутом **`id`**, содержащим код текстовки.

Пример:

```
<help id="msg.help">
```

В данном примере «msg.help» — код текстовки. Текстовки используются для локализации, подробно локализация описана в разделе 5.3.

В ТПО 7 версии в теге `<help>` возможно вывести минимальную и максимальную сумму платежа. Для этого используются атрибуты:

1) **`$min_sum_purchase`** — минимальная сумма покупки (совпадает с минимальной суммой по сервису);

2) **`$max_sum_purchase`** — максимальная сумма покупки (со своей комиссией не превышает максимальную сумму по сервису).

Пример:

```
<help>
      <![CDATA[Если хотите внести фиксированную сумму — укажите её.
                Можете оставить поле пустым.
                      Минимальная сумма: $min_sum_purchase
                      Максимальная сумма: $max_sum_purchase]]>
</help>
```

5.14 ВАЛИДАЦИЯ И ФИЛЬТРАЦИЯ ДАННЫХ

5.14.1 ПРОВЕРКА ВВЕДЕННЫХ ДАННЫХ (<VALIDATOR>)

Для локальной проверки введенных данных используется элемент, описываемый тегом `<validator>`. Описание типов валидаторов приведено в таблице 5.14.1.1.

Таблица 5.14.1.1 — Типы валидаторов

Тип	Описание	Пример

regex	Используется по умолчанию. Осуществляет проверку на соответствие регулярным выражениям. Дочерние элементы:
	• `<rule>` — в атрибуте **regex** описывает разрешающее правило;
	• `<block>` — в атрибуте **regex** описывает запрещающее правило.
	Данные будут валидны, если они удовлетворяют хотя бы одному разрешающему правилу и при этом не удовлетворяют всем запрещающим правилам
		`<validator type="regex">`
		`  <rules>`
		`  <rule regex="^12.*$"/>`
		`  ...`
		`  <rule .../>`
		`  <block regex="^.*7$"/>`
		`  ...`
		`  <block .../>`
		`  </rules>`
		`</validator>`

range	Осуществляется проверка на вхождение в заданный	`<validator type="range">` `  <range begin="0"`

Тип	Описание	Пример
	интервал. Недоступно для <text-field>	end="9999999.99"/> </validator>
cbc	Валидатор номеров карт сдачи. Возвращает **true**, когда введен верный номер карты сдачи. Поддерживаются 16-значные пин-коды	<validator type="cbc"/>
voucher	Валидатор ваучеров. Возвращает **true**, когда введен 12-ти значный пин	<validator type="voucher"/>
or	Логический валидатор, возвращает **true**, если хотя бы один вложенный валидатор вернул **true**	<validator type="or"> <validator type="regex"> <rules> <rule regex="^2.*"/> </rules> </validator> <validator type="regex"> <rules> <rule regex="^1.*"/> </rules> </validator> </validator>
and	Логический валидатор, возвращает **true**, если все вложенные валидаторы вернул **true**	<validator type="and"> <validator type="regex"> <rules> <rule regex="^2.*"/> </rules> </validator> <validator type="regex"> <rules> <rule regex="^1.*"/> </rules> /validator> </validator>
groovy	Осуществляется проверка в соответствии с groovy-скриптом, находящимся во внешнем файле. Исключен из	<validator type="groovy" name="имя файла"/>

Тип	Описание	Пример
	новой модели, по причине того, что его невозможно использовать на мобильных платформах. В качестве альтернативы используется js-валидатор	
luhn	Валидатор Луна, используется для контроля правильности ввода PAN (номера пластиковой карты)	`<validator type="luhn"/>`
yandex	Валидатор номера кошелька Яндекс.деньги. В 7 версии ТПО не поддерживается	`<validator type="yandex"/>`
custom	Любой внешний валидатор, тип внешнего валидатора определяется атрибутом **name**	`<validator type="custom" name="имя валидатора"/>`
js	Валидатор, позволяющий использовать для проверки данных **javaScript**	`<validator type="js"> <![CDATA[код на JavaScript]]> </validator>`
empty	Валидатор, позволяющий либо оставлять поле пустым, либо проверять его на соответствие вложенному валидатору	`<validator type="or"> <validator type=”empty”/> <validator type=”luhn”/> </validator>`
mod11	Валидатор по модулю 11	`<validator type="mod11"/>`
ibaniso7064	Валидатор номеров банковских счетов **IBAN** (*International Bank Account Number*), реализованный в соответствии со стандартом **ISO 7064**	`<validator type="ibaniso7064"/>`

В зависимости от типа валидатора доступны различные дочерние теги и атрибуты.

Примечание!
Внешний валидатор — это валидатор, который расположен не в библиотеке *input-model.jar*, а какой либо другой. Так, например, для использования js-валидатора необходимо присутствие библиотеки *js.jar*.

Дочерние элементы <validator> с type="regex": <rules> — список правил валидации.

Дочерние элементы <rules>: <rule> — определяет правильность ввода, <block> — определяет невалидный ввод.

Например, запись:

```
<rule regex="1.*"/>
```

говорит о том, что номер должен начинаться с «1», но

```
<block regex="1.*"/>
```

запрещает вводить все строки, которые начинаются с символа «1».

Валидатор с типом **regex** использует язык регулярных выражений. Используется локальной проверки введенных данных.

При объявлении используют элементы <rules>, <rule>, <block>.

Элемент <validator> с типом **regex** для проверки введенного значения с помощью регулярного выражения:

```
<validator type="regex">
      <rules>
          <rule regex=<notEmptyString>/>
      </rules>
</validator>
```

Правило задает условие на введенное значение.

Дочерние теги тега <validator> с type=«range»: <rules>.

Атрибуты тега `<validator>` с **`type=«range»`** приведены в таблице 5.14.1.2.

Таблица 5.14.1.2 — Атрибуты валидатора с типом **`«range»`**

Атрибут	Описание	Обяз.
`begin`	Начальное значение для проверки вводимого	Да
`end`	Конечное значение для проверки вводимого	Да

Пример использования валидатора с **`type=«js»`:**

В сценарии:

```
<validator name="js" type="js" >
   <![CDATA[
      !!!Валидатор!!!
   ]]>
</validator>
```

Примеры валидаторов:

В следующем примере определяется функция, внутри которой сначала происходит проверка длины number — это вводимое в поле значение, в самом поле атрибут может называться как угодно id1/number/phone и другие, но в функции вводимый номер ВСЕГДА number. Если длина number равна 13, то вычисляется значение m путем сложения произведения каждого символа номера и заданного числа. Затем определяется значение переменной c — остаток от деления m на 11. Переменной k присваивается разность между 11 и c. Если k равняется 11 или 10, то присваивается значение 0. Затем возвращается k, равное первым 12 символам number. Если обработка скрипта завершается успешно (не false), то номер считается валидным.

```
function f(number) {
  if (number.length == 13) {
                                var m = number[0] * 7 + number[1] * 6 + number[2] *
                                5 + number[3] * 4 + number[4] * 3 + number[5] * 2 +
                              number[6] * 7 + number[7] * 6 + number[8] * 5 +
                              number[9] * 4 + number[10] * 3 + number[11] * 2;
                              var c = m % 11;
                              var k = 11 - c;
                              if (k == 11 || k == 10) {
```

```
                                                    k = 0;
                                                    }
                              return k == number[12];
                             }
  return false;
}
f(number);
```

Валидные номера: 2709978450144,2204991445007.

```
function f(number) {
                      if (number.length != 22) {
                                                  return false;
                                                 }
                      var result = 0;
                      var count = 2;
                      for (j = 20; j >= 0; j--) {
                                                  result += count * number[j];
                                                  count += 1;
                                                  if (count > 7) {
                                                                  count = 2;
                                                                  }
                                                  }
                      step2 = result % 11;
                      var check = number[21];
                      if (step2 == 0 && step2 == check) {
                                                           return true;
                                                           }
                      step2 = 11 - step2;
                      if (step2 > 9 && step2 % 10 == check) {
                                                              return true;
                                                             }
                    return step2 == number[21];
}
f(number)
```

Валидные номера: 9100108510009616090595, 9100111510003038743850.

5.14.2 ФОРМАТИРОВАНИЕ ДАННЫХ (<DATA-FORMATTER>)

Для форматирования данных используется элемент, описываемый тегом `<data-formatter>`. Используется только для текстового поля на экране «NUMERIC». Работает только по нажатию кнопки «ДАЛЕЕ». Существуют два типа элемента:

1. **type=«regex»** — осуществляется проверка на соответствие регулярному выражению. Регулярные выражения задаются в дочерних элементах `<rules>`. Атрибуты элемента `<rules>` приведены в таблице 5.14.2.1.

2. **type=«simple»** — форматирование согласно маске, указанной в атрибуте **mask**:

   1) **mask=<string>** — описывает маску ввода. Символ «d» в маске означает, что в данной позиции разрешены только цифры, «c» — буквы, «*» — любые символы. Подробное описание правил составления масок приведено по адресу http://docs.oracle.com/javase/7/docs/api/javax/swing/text/MaskFormatter.html.

Таблица 5.14.2.1 — Описание атрибутов элемента `<rules>` `<data-formatter>` с типом **«regex»**

Атрибут	Описание	Обяз.
default	Задает правило разбивки вводимых данных по умолчанию	Нет
by-step	Указывает, что маска в текстовом поле будет отображаться по мере ввода символов	Нет

Пример:

```
<data-formatter type="regex">
      <rules default="**** **** **** ****" by-step="true"/>
</data-formatter>
```

5.14.3 ФОРМАТИРОВАНИЕ ЗНАЧЕНИЯ ПРИ ОТОБРАЖЕНИИ (<FORMATTER>)

Для форматирования вводимого значения при отображении используется элемент, описываемый тегом `<formatter>`. Типы элемента, дочерние элементы и атрибуты аналогичны `<data-formatter>`, раздел 5.14.2.

Элемент `<rule>` определяет правило разбивки введенных данных. В случае, если введённое значение соответствует регулярному выражению, заданному в атрибуте **`regex`**, то применяется правило разбивки, указанное в **`value`**.

Дочерние теги: отсутствуют.

Описание атрибутов тега `<rule>` приведено в таблице 5.14.3.1.

Таблица 5.14.3.1 — Описание атрибутов тега `<rule>`

Атрибут	Описание	Обяз.
`regex`	Регулярное выражение, на соответствие которому осуществляется проверка	Да
`value`	Правило разбивки данных	Да

В примере ниже введенные данные разбиваются на 4 группы по 4 символа.

Пример:

```
<formatter type="regex">
      <rules default="**** **** **** ****"/>
</formatter>
```

В примере ниже введенные данные разбиваются на 4 группы по 4 символа. Маска будет отображаться по мере ввода символов.

Пример:

```
<formatter type="regex">
      <rules default="**** **** **** ****" by-step="true"/>
</formatter>
```

В примере ниже введенные данные разбиваются на группы в зависимости от количества введенных символов по мере ввода символов.

Пример:

```
<formatter type="regex">
      <rules default="**** **** **** ****" by-step="true">
          <rule regex="^$" value="____ ____ ____ ____"/>
          <rule regex="^\d{1}$" value="*___ ____ ____ ____"/>
          <rule regex="^\d{2}$" value="**__ ____ ____ ____"/>
          <rule regex="^\d{3}$" value="***_ ____ ____ ____"/>
          <rule regex="^\d{4}$" value="**** ____ ____ ____"/>
          <rule regex="^\d{5}$" value="**** *___ ____ ____"/>
          <rule regex="^\d{6}$" value="**** **__ ____ ____"/>
          <rule regex="^\d{7}$" value="**** ***_ ____ ____"/>
          <rule regex="^\d{8}$" value="**** **** ____ ____"/>
          <rule regex="^\d{9}$" value="**** **** *___ ____"/>
          <rule regex="^\d{10}$" value="**** **** **__ ____"/>
          <rule regex="^\d{11}$" value="**** **** ***_ ____"/>
          <rule regex="^\d{12}$" value="**** **** **** ____"/>
          <rule regex="^\d{13}$" value="**** **** **** *___"/>
          <rule regex="^\d{14}$" value="**** **** **** **__"/>
          <rule regex="^\d{15}$" value="**** **** **** ***_"/>
          <rule regex="^\d{16}$" value="**** **** **** ****"/>
      </rules>
</formatter>
```

Внимание!
Программное обеспечение «Платформа электронных кошельков SmartKeeper» не поддерживает возможность указания нескольких масок, т. е. вложенных элементов `<rule>`.

5.14.4 ИЗМЕНЕНИЕ ЗНАЧЕНИЯ (<MODIFICATOR>)

Для изменения введенного значения с помощью регулярного выражения используется элемент, описываемый тегом `<modificator>`.

Структура элемента:

```
<modificator type=[regex]>
      <rules>
          <rule regex=<notEmptyString>
                target=<notEmptyString>
                value=<notEmptyString>/>
      </rules>
</modificator>
```

Дочерние элементы и атрибуты аналогичны `<replaces>`, используемому в формах — подробно описан в документе «Формы оплаты для универсального модуля ввода данных. Программное обеспечение «Процессинговый центр Pay-logic». Руководство администратора». Поведение `<modificator>` при отсутствии **`target`** аналогично поведению `<replaces>`, то есть атрибут **`target`** не является обязательным, в качестве него используется значение **`regex`**.

Пример:

```
<modificator type="regex">
      <rules>
          <rule regex="^\d{8}$" target="^(\d{2})(\d{2})(\d{4})$"
                value="$1.$2.$3"/>
      </rules>
</modificator>
```

В строчку из 8 цифр добавляем точки между 2 и 3 символами и между 4 и 5 символами (из даты формата ddmmyyyy получили dd.mm.yyyy). Правило задает формат передаваемых данных (значение введено цифрами без дополнительных символов, а передается с дополнительными символами).

5.14.5 ФИЛЬТРАЦИЯ ДАННЫХ (<FILTER>)

Определить какие символы допустимы для ввода в какое-либо поле возможно с использованием правил фильтраций. Для указания правил фильтрации данных используется элемент, описываемый тегом `<filter>`. Элемент аналогичен фильтру, используемому в формах — подробно описан в документе «Формы оплаты для универсального модуля ввода данных. Программное обеспечение «Процессинговый центр Pay-logic». Руководство администратора».

6 ДЕЙСТВИЯ (ЭЛЕМЕНТЫ <ACTION>)

6.1 УСЛОВИЯ, ЦИКЛЫ, ВЕТВЛЕНИЯ

6.1.1 ЦИКЛИЧЕСКИЙ ОПЕРАТОР (<FOR>)

Действие `<for>` осуществляет циклический проход. Существуют два варианта использования: цикл перебора элементов и цикл с использованием счетчика. Доступно только в 5, 7 версии ТПО. Структура действия:

1 вариант:

```
<for key=<keyType>>
      ...
</for>
```

`key` — содержит ключ атрибута, по которому осуществляется циклический проход.

2 вариант:

```
<for from=<integer>
     to=<integer>
     step=<integer>>
       ...
</for>
```

Дочерние элементы: отсутствуют. **Атрибуты** приведены в таблице 6.1.1.1.

Таблица 6.1.1.1 — Атрибуты действия `<for>`

Атрибут	Описание	Обяз.
`from`	Начальная позиция цикла	Нет
`to`	Конечная позиция цикла	Нет
`step`	Шаг цикла	Нет

Атрибуты **key** и **from/to/step** — взаимоисключающие конструкции.

При использовании 1 варианта **key** содержит идентификатор элемента **#for-element** — текущий элемент.

Пример:

```
<for key="#services">
      <unpack type="data" key="#for-element" elements="saldo"/>
      <modify src-key="saldo" t-key="^(.*)$" t-value="^[-]{1}(.*$)"
          t-value-title="^[-]{1}(.*$)" v-key="summ" v-value="$1"
          v-value-title="$1"/>
      <pack type="data" key="#for-element" elements="saldo,summ"/>
      <clear keys="summ,saldo"/>
</for>
```

При использовании второго варианта атрибуты **from** и **to** определяют начальное и конечное значения границ перебора соответственно, **step** — шаг перебора, необязательный параметр, по умолчанию равен 1. В качестве значения атрибута **to** возможно указать переменную, используя синтаксис **${ключ переменной}**. Цикл выполняется до тех пор пока счетчик цикла меньше значения, заданного в атрибуте **to**. Во время работы цикла в контекст кладется переменная **for-index** (номер текущего элемента), которую возможно использовать для получения новой переменной путем склеивания одной переменной и порядкового номера **for-index**.

В примере ниже цикл повторяется от 0 до введенного на экране **number** значения. Переменная **zp.${#for-index}** (zp.0, zp.1, zp.2, zp.3 и т. д.) сравнивается с 1. Если переменная равна 1, то осуществляется переход на экран оплаты.

Пример:

```
<scenario begin="input_info">
 <!--Создание экрана ввода числовой и текстовой информации-->
 <screen type="numeric" title="Введите кол-во повторений цикла"
         id="input_info">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 2. Название поля
ввода: "Кол-во повторений цикла"-->
```

```
    <text-field id="number" title="Кол-во повторений цикла" max-len="2"
                keyboard="Digital">
     <!--Регулярное выражение для валидации вводимого кол-ва
повторений-->
     <validator type="regex">
      <rules>
       <rule regex="^\d{1,2}$" />
      </rules>
     </validator>
    </text-field>
   </fields>
   <actions>
    <!--Описание поведения сценария оплаты при нажатии "Далее"-->
    <action type="Next" title="Далее">
     <!--Объявление переменной-->
     <set key="zp.0" key-title="Zp" value="0" value-title="0"/>
     <set key="zp.1" key-title="Zp" value="0" value-title="0"/>
     <set key="zp.2" key-title="Zp" value="1" value-title="1"/>
     <set key="zp.3" key-title="Zp" value="3" value-title="3"/>
     <set key="zp" key-title="Zp" value="0" value-title="0"/>
     <for from="0" to="${number}" step="1">
      <if condition="zp.${#for-index} == 1">
       <then>
        <goto target="pay"/>
        <cancel/>
       </then>
      </if>
      </for>
    </action>
    <!--Описание поведения сценария оплаты при нажатии "Назад"-->
    <action type="Prev" title="Назад">
     <goto target="previous" />
    </action>
    <!--Описание поведения сценария оплаты при нажатии "Выход"-->
    <action type="Exit" title="Выход">
     <goto target="exit"/>
    </action>
   </actions>
  </screen>
</scenario>
```

6.1.2 ОПЕРАТОРЫ УСЛОВНОГО ПЕРЕХОДА (<IF>, <SWITCH>)

Для формирования условий выполнения того или иного действия используется оператор `<if>`. Всегда содержит дочерний элемент `<then>` и при необходимости `<else>`, которые не содержат атрибутов. Возможны два варианта использования:

1. С атрибутом **`expression`** — используется синтаксис JavaScript (ТПО 5, 7).

Структура элемента:

```
<if expression=<notEmptyString40>>
  <then> ... </then>
  <else> ... </else>
</if>
```

2. С атрибутом **`condition`** — логическое сравнение. Возможные значения:

1) **`==`** — равно, можно сравнивать числовые и строковые переменные;

2) **`!=`** — не равно, можно сравнивать числовые и строковые переменные;

3) **`gt`** — больше;

4) **`ge`** — больше либо равно;

5) **`lt`** — меньше;

6) **`le`** — меньше либо равно;

7) **`~`** — эквивалентно, используется в сравнении с регулярным выражением, например, condition="SUM ~ ^0$". Для проверки на пустое значение используйте регулярное выражение "^$";

8) **`is-not-null`** — проверка на значение не равное **`null`**, например, **`condition=«is-not-null fio»;`**

9) **is-null** — проверка на значение равное **null**, например, **condition=«is-null fio»**.

Структура оператора:

```
<if condition=[==|!=|gt|ge|lt|le|~|is-not-null|is-null]>
  <then> ... </then>
  <else> ... </else>
</if>
```

Дочерние элементы: <then>, <else>.

Наличие элемента <else> не является обязательным, но лучше его вводить и сразу закрывать <else/>.

Пример:

```
<if condition="sum gt 0">
  <then>
    <set-sum from="sum" type="units"/>
      <goto target="pay"/>
  </then>
  <else>
    <goto target="input_sum"/>
  </else>
</if>
```

В примере сравнивается значение переменной **sum** с нулем, и если она больше, то ее значение передается в сумму платежа и осуществляется переход на экран оплаты, иначе — на экран ввода суммы.

```
<if expression="id1 > (24 + param)^2 || Math.round(id1/3) < x">
      <then/>
      <else/>
</if>
```

Оператор множественного выбора (<switch>) позволяет выполнять различные инструкции в зависимости от условий. Существуют 3 формата записи:

1. Значение атрибута **value** элемента <case> проверяется на равенство со значением атрибута **value** элемента <switch> и выполняются инструкции в

соответствии с <case>. Поддерживается ПО «Платформа электронных кошельков SmartKeeper».

Пример:

```
<switch value="id1">
      <case value="10"> ... </case>
      <case value="15"> ... </case>
      <default/>
</switch>
```

2. Значение атрибута **value** элемента <case> проверяется на равенство с результатом вычисления атрибута **expression** элемента <switch> и выполняются инструкции в соответствии с <case>. ПО «Платформа электронных кошельков SmartKeeper» вариант использования не поддерживается.

Пример:

```
<switch expression="id1 % 10">
      <case value="10"> ... </case>
      <case value="15"> ... </case>
      <default/>
</switch>
```

3. В следующем <switch> выполнится первый <case>, чье значение выражения **expression** истинно (**true**) или **default**, если таковых не было, и выполняются инструкции в соответствии с <case>. ПО «Платформа электронных кошельков SmartKeeper» вариант использования не поддерживается.

Пример:

```
<switch>
      <case expression="id1 &lt; 10"> ... </case>
      <case expression="id1 == 24"> ... </case>
      <default/>
</switch>
```

Выражение **expression** элемента <switch> позволяет использовать математические операции:

1. Math.abs() — возвращает абсолютное (положительное) значение переменной.

2. `Math.max()` — выбирается максимальное значение.

3. `Math.min()` — выбирается минимальное значение.

4. `Math.round()` — осуществляет математическое округление.

Пример:

```
<switch expression="Math.max(id1,id2)">
   <case value="1">
      ...
   </case>
      ...
</switch>
```

Выражение **`expression`** элемента `<case>` позволяет использовать функции:

1. `null` — проверка значения переменной на null.

2. `undefined` — проверка переменной на существование.

Пример:

```
<case expression="typeof tek_pok_noch == &amp;quot;undefined&amp;quot;">
   ...
</case>
```

6.1.3 ОПЕРАТОР БЕЗУСЛОВНОГО ПЕРЕХОДА (<GOTO>)

Для перехода на указанный экран используется действие `<goto>`. Структура действия:

```
<goto target=[exit|pay|<string>]
      target-else=<screenNameType>
      expression=<screenNameType>/>
```

Атрибуты приведены в таблице 6.1.3.1.

Таблица 6.1.3.1 — Атрибуты действия `<goto>`

Атрибут	Описание	Обяз.
target	Указывает на какой экран требуется переход. Обязательный атрибут. Возможные значения: **exit** — выход в главное меню, **pay** — переход на экран оплаты, **<notEmptyString40>** — идентификатор экрана, на который осуществляется переход	Да
target-else	Идентификатор экрана, на который осуществляется переход	Нет
expression	Условие, при выполнении которого будет осуществлен переход на экран, указанный в атрибуте **target**, при невыполнении — **target-else**	Нет

Пример:

```
<goto target="second_screen"/>
```

Переход на экран **«second_screen»**.

```
<goto expression="count > 100" target="screen" target-else="screen2"/>
```

Переход на экран **«screen»**, если **«count > 100»**, иначе переход на экран **«screen2»**.

6.2 ОТМЕНА (<CANCEL>) И ОЧИСТКА ПЕРЕМЕННЫХ (<CLEAR>)

В случае, если необходимо остаться на данном экране, например, после проверки правильности ввода номера, используется действие <cancel>:

```
<cancel/>
```

Дочерние элементы: отсутствуют.

Атрибуты: отсутствуют.

Пример:

```
<if condition="n gt 5" >
   <then>
     <dialog type="Error" title="Error"
             message="The total number of tickets may not exceed 10!"
             timeout="10" default="okay">
        <actions>
          <action type="okay" title="OK">
            <cancel/>
          </action>
        </actions>
     </dialog>
   </then>
</if>
```

Для удаления инициализированных переменных используются действия:

1. <clear-all> — используется в случае, если необходимо удалить все ранее инициализированные переменные:

```
<clear-all/>
```

2. <clear> — удаляет переменные из контекста:

```
<clear keys=<keyListType>/>
```

3. `<clear-except>` — удаляет значения всех инициализированных в сценарии переменных за исключением, указанных в атрибутах элемента:

```
<clear-except keys=<keyListType>/>
```

4. `<clear-like>` — служит для удаления переменных из контекста, соответствующих регулярному выражению:

```
<clear-like regex=<notEmptyString40>/>
```

Действие `<clear-all>` используется без указания атрибутов, действия `<clear>`,`<clear-except>` имеют атрибут **`keys`** — переменные перечисляются через запятую без пробелов. Обязательный атрибут. Непустая строка до 30 символов. Действие `<clear-like>` имеет атрибут **`regex`** — регулярное выражение, которому соответствуют удаляемые переменные. Более подробное описание регулярных выражений приведено по данному адресу.

Примеры:

```
<action type="navi1" title="Select Trip Type">
      <clear-all/>
      <goto target="get_sectors"/>
</action>
```

Удаляет все ранее инициализированные переменные.

```
<clear keys="date,sum"/>
```

Удаляет переменные **`date`** и **`sum`**.

```
<clear-except keys="date,sum"/>
```

Удаляет все переменные за исключением **`date`** и **`sum`**.

```
<clear-like regex="^sum*$"/>
```

Удаляет все атрибуты **`sum`**, **`summ`**, **`summm`** и т.д., так как «*» относится к «m».

```
<clear-like regex="^sum.*$"/>
```

Подразумевает что в названии есть **`sum`** и далее любые символы в количестве от нуля и более. Удаляет все атрибуты **`sum`**, **`summ`**, **`sum1`**, **`sum_first`** и т.д.

```
<clear-like regex="^name.*$"/>
```

Удалят все атрибуты, начинающиеся на name: **name1**, **name2**, **name_first** и т. д.

```
<clear-like regex="^serviceid.*$"/>
```

Удалят все атрибуты, начинающиеся на serviceid: **serviceid1**, **serviceid2**, **serviceid_first** и т. д.

6.3 МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ <MATH>

6.3.1 ОБЩИЕ СВЕДЕНИЯ

Существуют два варианта использования элемента `<math>`: с использованием атрибута **`operation`** и с использованием атрибута **`expression`**.

6.3.2 ИСПОЛЬЗОВАНИЕ <MATH> С АТРИБУТОМ `OPERATION`

Действие `<math>` с атрибутом **`operation`** служит для выполнения простых математических действий с числовыми переменными: сложение, вычитание, умножение, деление. Используются переменные с плавающей точкой типа **`double`**.

Структура элемента:

```
<math operation=<notEmptyString40>
      result-key=<notEmptyString25>
      result-title=<notEmptyString25>
      flags=<flags_specification>/>
```

Атрибуты описаны в таблице 6.3.2.1.

Таблица 6.3.2.1 — Атрибуты действия `<math>` с атрибутом **`operation`**

Атрибут	Описание	Обяз.
`operation`	Математическое выражение включающее до двух переменных, например, **«10 * 10»**, **«sum / 100»**, **«sum + com»**. Между переменными и знаком должен стоять	Да

Атрибут	Описание	Обяз.
	пробел	
result-key	Переменная, которой присваивается получаемое значение	Нет
result-title	Название переменной, которой присваивается получаемое значение	Нет
flags	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет

Позволяет выполнять только одну математическую операцию, то есть невозможно выполнить действие типа «a ? b ? c ? ...», где «?» — любая из выше перечисленных операций.

Пример:

```
<math operation="sum + com"
      result-key="total"
      result-title="Сумма с учетом комиссии"/>
```

В данном примере в переменную **total** попадает значение из суммы переменных **sum** и **com**.

6.3.3 ИСПОЛЬЗОВАНИЕ <MATH> С АТРИБУТОМ EXPRESSION

Действие <math> с использованием атрибута **expression** служит для выполнения математических действий с числовыми переменными любой сложности.

Структура действия:

```
<math expression=[+|-|*|div|mod|^|round()|abs()|max()|min()]
      result-key=<notEmptyString25>
      result-title=<notEmptyString25>
      flags=<flags_specification>/>
```

Атрибут flags описан в разделе 5.5, атрибуты **result-key**, **result-title** описаны в таблице 6.3.2.1. Атрибут **expression** — математическое выражение любой сложности, например, **«@d3 — summ * skidka»**, **«(total mod 10) + summ * kolvo»**. Между переменными и знаком должен стоять пробел. Доступные операции:

1. **+** — сложение.
2. **-** — вычитание.
3. ***** — умножение.
4. **/** — деление.
5. **div** — целочисленное деление.
6. **%** — получение остатка от деления.
7. **^** — возведение в степень.
8. **Math.round()** — округление числа. Например, «Math.round(id13)».
9. **Math.abs()** — получение модуля числа. Например, «Math.abs(id13)».
10. **Math.max()** — поиск максимального значения. Например, «Math.max(id13,id18,8)».

11. **`Math.min()`** — поиск минимального значения. Например, «Math.min(id13,id18,8)».

Пример:

```
<math expression="2 * (id2 % 3) ^ 3"
      result-key="id4"
      result-title="Сумма"/>
```

В данном примере мы находим остаток от деления переменной **`id2`** на 3, возводим полученное значение в третью степень и умножаем на 2. Получившийся результат записываем в переменную **`id4`**.

6.4 ПРЕОБРАЗОВАНИЯ ЗНАЧЕНИЙ АТРИБУТОВ

6.4.1 КОНВЕРТАЦИЯ ЗНАЧЕНИЙ АТРИБУТОВ (<CONVERT>)

Для конвертации значения атрибута из одной системы счисления в другую используется действие `<convert>`. Структура действия:

```
<convert key=<notEmptyString15>
         from=<integer>
         to=<integer>/>
```

Дочерние элементы: отсутствуют.

Атрибуты приведены в таблице 6.4.1.1.

Таблица 6.4.1.1 — Атрибуты действия `<convert>`

Атрибут	Описание	Обяз.
key	Переменная, которую требуется конвертировать	Да
from	Разрядность начальной системы счисления	Да
to	Разрядность конечной системы счисления	Да

Пример:

```
<convert key="id2" from="10" to="2"/>
```

Конвертирует значение атрибута **id2** из десятичной в двоичную систему счисления.

6.4.2 ПРЕОБРАЗОВАНИЕ СТРОКОВЫХ ПЕРЕМЕННЫХ (<MODIFY>)

Для преобразования строковых переменных используется действие `<modify>`. С его помощью можно обрезать строку, добавить в нее символы, преобразовать значение переменной и др.

```
<modify src-key=<keyType>
        t-key=<notEmptyString>
        t-key-title=<notEmptyString>
        t-value=<notEmptyString>
        t-value-title=<notEmptyString>
        t-original-value=<notEmptyString>
        v-key=<notEmptyString>
        v-key-title=<notEmptyString>
        v-value=<notEmptyString>
        v-value-title=<notEmptyString>
        v-original-value=<notEmptyString>
        flags=<flags_specification>
        flags-action=[add|remove]
        v-key-title-id=<resourceIdType>
        v-value-title-id=<resourceIdType>/>
```

Атрибуты описаны в таблице 6.4.2.1.

Таблица 6.4.2.1 — Атрибуты действия `<modify>`

Атрибут	Описание	Обяз.
src-key	Ключ переменной, из которой берется значение	Да
t-key	Регулярное выражение, на соответствие которому осуществляется проверка значения переменной	Нет
t-key-title	Регулярное выражение, на соответствие которому осуществляется проверка названия переменной	Нет
t-value	Регулярное выражение, с помощью которого выбирается реальное значение переменной	Нет
t-value-title	Регулярное выражение, с помощью которого	Нет

Атрибут	Описание	Обяз.
	выбирается отображаемое значение переменной	
t-original-value	Оригинальное значение переменной	Нет
v-key	Ключ создаваемой переменной. Если не указан, значение будет перезаписано в оригинальную переменную (**src-key**)	Нет
v-key-title	Название создаваемой переменной	Нет
v-value	Значение создаваемой переменной	Нет
v-value-title	Отображаемое значение создаваемой переменной	Нет
v-original-value	Оригинальное значение создаваемой переменной	Нет
flags	Параметр работает совместно с параметром **flags-action**. Если задано значение **add** параметра **flags-action**, то для изменяемого в <modify> параметра заданные в параметре **flags** флаги будут добавлены. Если задано значение **remove** параметра **flags-action**, то для изменяемого в <modify> параметра заданные в параметре **flags** флаги будут удалены. Возможные значения параметра **flags** описаны в разделе 5.5. Флаги могут задаваться как числом (в 5 и 7 версиях ТПО), так и символьными константами (только в 7 версии ТПО)	Нет
flags-action	Заданные в параметре **flags** флаги будут установлены или сняты для изменяемого в <modify> параметра в зависимости от значения **flags-action**. Если задано значение **add** — флаги будут добавлены, **remove** — флаги будут удалены	Нет
v-key-title-id	Значение ключа, по которому в ресурсах хранится текстовка, которая будет подставлена в **v-key-title** при отправке атрибута платежа на сервер. Текстовки расположены в каталоге ***<корень***	Нет

Атрибут	Описание	Обяз.
	ТПО>/res/module/i18n/ в файле *input.properties*. Поддерживается локализация, например, файл с текстовками на английском языке будет называться *input_en.properties*. Доступен в ТПО 7, 5 версией ТПО не обрабатывается	
v-value-title-id	Значение ключа, по которому в ресурсах хранится текстовка, которая будет подставлена в **v-value-title** при отправке атрибута платежа на сервер. Текстовки расположены в каталоге ***<корень ТПО>/res/module/i18n/*** в файле *input.properties*. Поддерживается локализация, например, файл с текстовками на английском языке будет называться *input_en.properties*. Доступен в ТПО 7, 5 версией не обрабатывается	Нет

Пример:

```
<modify src-key="sum"
        t-key="^(.*)$"
        t-key-title="^(.*)$"
        t-value="^(\d{1,5})(\d{2})$"
        t-value-title="^(\d{1,5})(\d{2})$"
        v-key="sum1"
        v-key-title="Сумма"
        v-value="$1.$2"
        v-value-title="$1.$2"
        flags="0x02"
        flags-action="remove"/>
```

В данном примере переменной **sum1**, название которой «Сумма», присваивается значение, полученное из переменной **sum** согласно правилу «^(\d{1,5})(\d{2})$». Значение новой переменой (реальное и отображаемое) преобразуется по правилу «$1.$2». То есть, переменная **sum=»67105»** будет преобразована в переменную **sum1=«671.05»**. При этом флаг «Не выводить на печать в чеке» (**HIDE_ON_PRINT**) будет удален.

Внимание!
Действие <modify> не выполняется, если <**action type="error"**>.

6.4.3 ФОРМАТИРОВАНИЕ ДАТЫ (<DATE-UTIL>)

Для работы с форматом даты служит действие `<date-util>`, структура действия:

```
<date-util src-key=<notEmptyString15>
           target-key=<notEmptyString15>
           input-format=<notEmptyString25>
           output-format=<notEmptyString25>
           amount=<integer>
           field=[0|1|2|3|4|5|6|7|8|9|10|11|12|13|14]/>
```

Дочерние элементы: отсутствуют.

Атрибуты приведены в таблице 6.4.3.1.

Таблица 6.4.3.1 — Атрибуты действия `<date-util>`

Атрибут	Описание	Обяз.
src-key	Переменная, из которой берется значение	Да
target-key	Переменная, в которую записывается значение	Да
input-format	Входной формат значения (используется библиотека java.text.SimpleDateFormat)	Да
output-format	Выходной формат значения (используется библиотека java.text.SimpleDateFormat)	Да
amount	Сдвиг на указанное число величин, величина определяется в атрибуте **field**	Нет
field	Определяет с какими величинами осуществляется работа. Возможные значения: 0 — эра, 1 — год, 2 — месяц, 3 — неделя в году, 4 — неделя месяца, 5 — день месяца, 6 — день в году, 7 — день недели, 8 — день недели в месяце, 9 — часы в формате am/pm, 10 —	Нет

Атрибут	Описание	Обяз.
	часы в формате 24 часа, 11 — час дня, 12 — формат в минутах, 13 — формат в секундах, 14 — формат в милисекундах. Более подробно в описании библиотеки java.util.Calendar	

Пример:

```
<date-util src-key="now"
           target-key="now"
           input-format="yyMMdd"
           output-format="dd.MM.yyyy"
           amount="60"
           field="6"/>
```

Изменяем формат вывода значения переменной **now** с «**yyMMdd**» на «**dd.MM.yyyy**» со сдвигом на 60 дней вперед.

6.4.4 ЗАМЕНА СИМВОЛОВ (<ENCODE>)

Выполнить замену символов значения атрибута позволяет действие `<encode>`. Имя файла, хранящего таблицу замен, определяется в атрибуте **map** (таблица 6.4.4.1). Например, если **map=«file»**, то файл будет называться *file.properties*. В 5 версии ТПО файл размещается в каталоге ***<корень ТПО>/resources/encode***/, в 7 — ***<корень ТПО>/res/module/input/encode/***. Также реализована поддержка в РМА, каталог: ***<корень РМА>/resources/encode***/. Структура действия:

```
<encode keys=<notEmptyString>
        map=<notEmptyString15>/>
```

Дочерние элементы: отсутствуют.

Атрибуты приведены в таблице 6.4.4.1.

Таблица 6.4.4.1 — Атрибуты действия `<encode>`

Атрибут	Описание	Обяз.
keys	Переменная, значение которой требуется заменить. Может принимать список ключей для изменения	Да
map	Значение соответствующего атрибута	Да

Пример:

```
<encode keys="type1,type2,type3" map="cbar"/>
```

Пример содержимого файла cbar.properties:

```
Ü=U
Ö=O
Ğ=G
İ=I
Ə=A
Ç=C
Ş=S
1=1
2=2
3=3
4=4
```

До знака «=» указывается символ, который требуется заменить, после — значение, на которое символ заменяется. Если символ отсутствует в файле, то он не изменяется. Примером использования может служить замена символов в считанном штрих-коде.

6.5 ИНТЕРАКТИВНЫЕ ДИАЛОГИ (<DIALOG>)

Для вывода диалогового окна на экране сервиса, обычно с предупреждающей информацией, используется действие `<dialog>`.

Структура действия:

```
<dialog type=[Info|Warning|Error|Question]
        title=<titleType>
        timeout=<nonNegativeInteger>
        default=<notEmptyString25>
        message=<dialogMessageType>
        mess-params=<keyListType>
        message-id=<resourceidType>
        title-id=<resourceidType>>
      <actions>
          <action type=[next|prev|exit|skip|redirect|edit|pay|previous]
                  title=<titleType>>
                ...
          </action>
      </actions>
</dialog>
```

Атрибуты описаны в таблице 6.5.1.

Таблица 6.5.1 — Атрибуты действия `<dialog>`

Атрибут	Описание	Обяз.
type	Внешний вид диалогового окна, зависит от реализации скина. Возможные значения: • **info** — на экране выводится информационной сообщение; • **warning** — на экране выводится предупреждение;	Да

Атрибут	Описание	Обяз.
	• **error** — на экране выводится сообщение об ошибке; • **question** — на экране выводится сообщение, ожидающее ответной реакции пользователя; • **selection** — диалог выбора варианта.	
title	Выводимый заголовок диалогового окна.	Да
timeout	Указывается время, на которое выводится диалоговое окно.	Нет
default	Значение, которое принимается по умолчанию. В default можно указать любое значение и оно будет обработано, если есть в сценарии. Если значения нет в сценарии, то возникнет NPE.	Нет
message	Используется для указания заголовка экрана ввода	Да
mess-params	Выводимые в сообщении значения переменных. Переменные перечисляются через запятую без пробелов.	Нет
title-id	Содержит идентификатор текстовки, которая будет подгружена в качестве значения **title**. Доступен только в ТПО 7.	Да
message-id	Идентификатор текстовки, которая будет использована в качестве сообщения заголовка экрана ввода. Доступен только в ТПО 7.	Нет

Для элемента `<dialog>` доступны следующие типы `<action>`:

1. **next** — переход далее.
2. **prev** — переход на предыдущий экран.
3. **exit** — выход в главное меню.

4. **skip** — пропуск экрана.

5. **redirect** — редирект на другой сервис.

6. **edit** — показ описания экрана.

7. **pay** — к оплате.

8. **previous** — переход в меню, откуда был осуществлён вход в сервис.

Для того чтобы значение переменной N выводилось в сообщении нужно в тексте сообщения указать «{N-1}», то есть для первой переменной будет указано «{0}». Элемент <dialog> представляет собой активную часть экрана в экране, поэтому он также использует элемент <actions> (элемент <fields> в нем отсутствует). Количество активных кнопок на форме диалога не больше двух (например, Ok или Да и Нет).

Пример 1 (рисунок 6.5.1):

```
<dialog type="Info" title="Внимание" timeout="10" default="okay"
        message="Введенный лицевой счет {0} не существует"
        mess-params="id1">
    <actions>
       <action type="okay" title="Ok"> <cancel/> </action>
    </actions>
</dialog>
```

В примере 1 в **default** указано значение **«okay»**, это же значение определено в <action>.

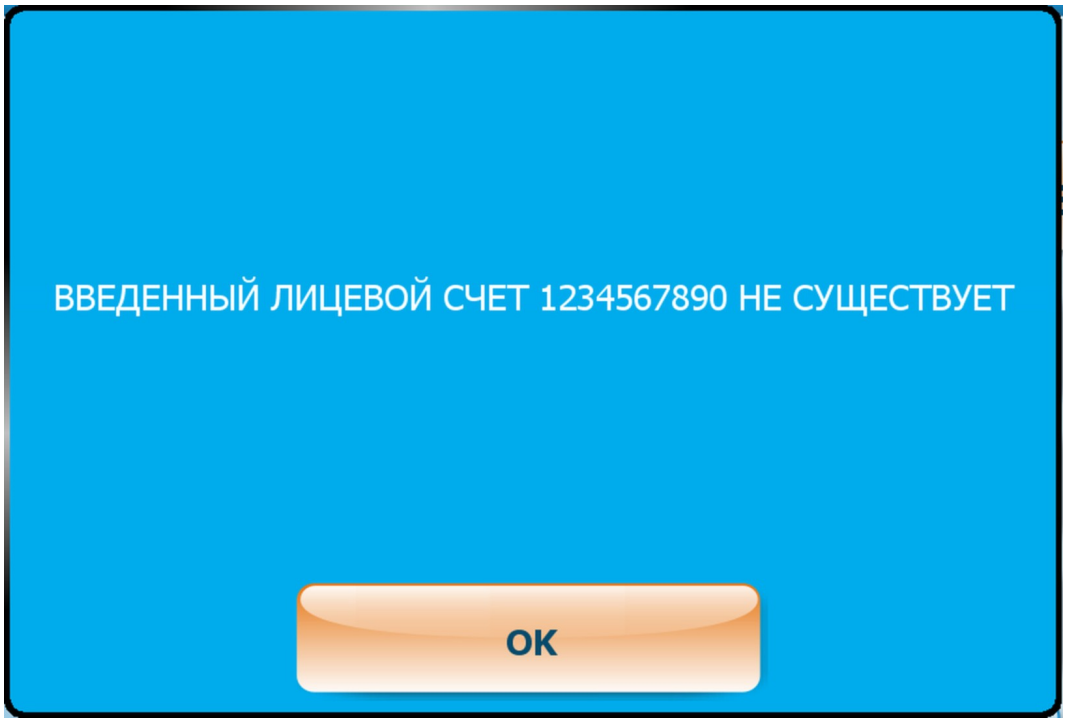


Рисунок 6.5.1 — Диалоговый экран с одной кнопкой

Пример 2 (рисунок 6.5.2):

```
<dialog type="Question" title="Внимание" timeout="10" default="no"
        message="Хотите оплатить пеню?" >
   <actions>
     <goto-action type="yes" title="ДА" target="with_fine"/>
     <goto-action type="no" title="НЕТ" target="without_fine"/>
   </actions>
</dialog>
```

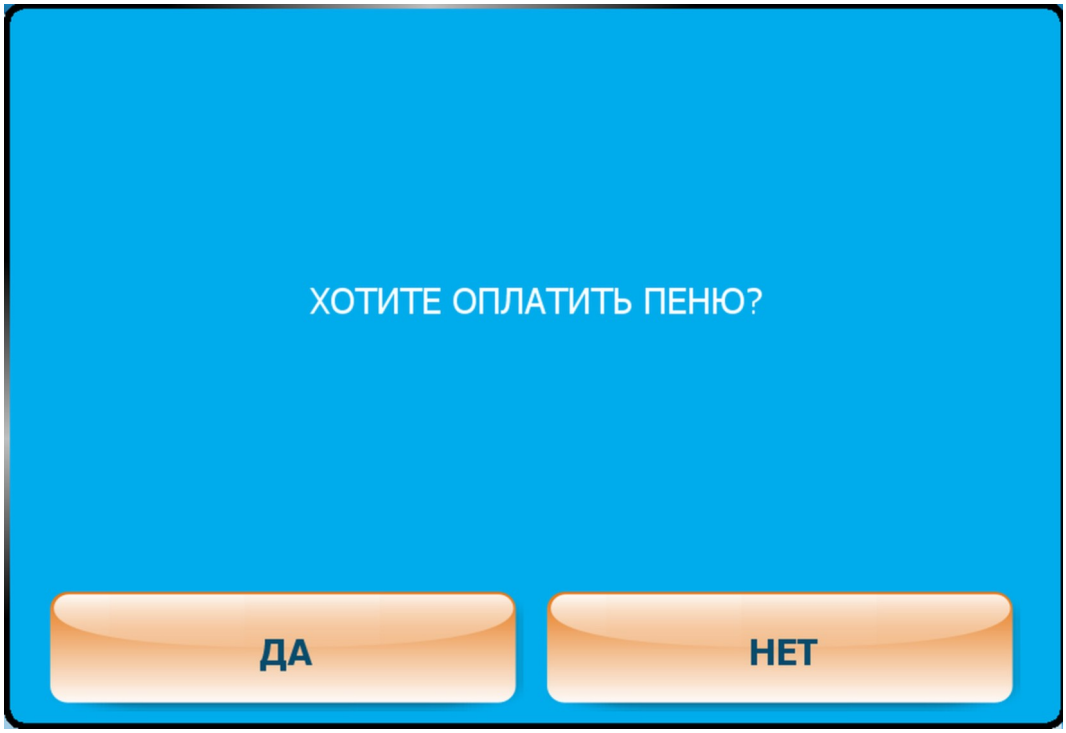


Рисунок 6.5.2 — Диалоговое окно с двумя кнопками

Имеется возможность стилизовать диалоговые окна при разработке/модификации интерфейса.

6.6 ОНЛАЙН-ЗАПРОС (<ONLINE-REQUEST>)

Для реализации онлайн-запросов из сценариев используется действие `<online-request>`. Онлайн-запрос представляет собой запрос к серверу для получения каких-либо данных, выполняемый в режиме реального времени. В онлайн-запросе на сервер передаются заданные параметры, а от сервера — в случае успешного ответа могут возвращаться атрибуты. Возвращаемые атрибуты помещаются в контекст сценария и после выполнения онлайн-запроса доступны так же, как и любые ранее определенные или созданные переменные. Параметры и возвращаемые атрибуты (их количество, имена и содержание) зависят от вызываемой функции и реализации шлюза, их следует уточнять у сотрудников технической поддержки. Примером результата запроса является вывод баланса по переданному в запросе номеру лицевого счета. Структура действия:

```
<online-request type=[default]
                function=<notEmptyString25>
                params=<keyListType>
                provider=<srvId>
                attempts=<nonNegativeInteger>
                timeout=<nonNegativeInteger>
                timeout-all=<nonNegativeInteger>>
      <actions>
          <action type=[success|error|exception]>
                ...
          </action>
                ...
      </actions>
</online-request>
```

Дочерние элементы: `<actions>`.

Атрибуты описаны в таблице 6.6.1.

Таблица 6.6.1 — Атрибуты действия `<online-request>`

Атрибут	Описание	Обяз.
`type`	Тип выполняемого запроса, в настоящее время реализован только один тип запроса — **`default`**	Да
`function`	Название функции, выполняемой на стороне сервера процессинга при выполнении запроса к серверу провайдера. Название функции необходимо уточнять у разработчика шлюза. По умолчанию используется функция `adv-check`, которая выполняет онлайн-проверку. Для запуска модулей предварительной проверки используются функции: • `adv-verify` — при вызове функции выполняется проверка согласно модулям предварительной проверки и если она пройдена, то выполняется онлайн-проверка через шлюз; • `adv-preliminary` — при вызове функции осуществляется проверка через модули предварительной проверки без обращения к провайдеру (без вызова проверки, реализованной в шлюзе). Модули предварительной проверки подробно описаны в документе «Программное обеспечение «Процессинговый центр Pay-logic». Руководство администратора». Проверка согласно модулям предварительной проверки автоматически не осуществляется. Если в шлюзе реализован один метод, то название функции возможно не указывать: **`function=""`**. Если провайдер возвращает комиссию поставщика и в шлюзе реализована поддержка запроса комиссии, то в	Да

Атрибут	Описание	Обяз.
	онлайн-запросе следует использовать функцию `get-fee`. В этом случае вместо верхней комиссии по сервису будет установлена фиксированная сумма комиссии от поставщика. Поддерживается в ТПО 5 и 7	
`params`	Передаваемые переменные при запросе, по которым необходимо вернуть информацию, перечисляются через запятую без пробелов. Должен быть указан хотя бы один параметр. Возможно также указание всех имеющихся переменных без перечисления через $all Если провайдер возвращает комиссию поставщика и в шлюзе реализована поддержка запроса комиссии, то в онлайн-запросе с функцией `get-fee` следует передавать параметры «**`amount-without-fee,fee-calculation-method`**»	Да
`provider`	Для данного атрибута явно указывается через какого провайдера будет осуществляться онлайн-запрос	Нет
`attempts`	Количество попыток выполнения	Нет
`timeout`	Задержка между запросами, в секундах	Нет
`timeout-all`	Суммарное время выполнения запроса, в секундах	Нет
`reserve-id`	Принудительный запрос ID операции до совершения оплаты. Возможные значения: **`true, false`**. Используется в случае, когда в параметрах точек выключен параметр **«Получение ID операции перед созданием платежа на терминале»** (подробнее см. в документе «Кабинет агента»), однако в рамках определенного сервиса требуется отправлять зарезервированный ID провайдеру. Указывается только совместно с атрибутом **`id-operation-param-name`** (описан ниже), в котором будет храниться значение ID. Обрабатывается в ТПО 5,	Нет

Атрибут	Описание	Обяз.
	7 с версий x.121. **Пример:** `<online-request type="default" function="add-verify" params="id1" reserve-id="true" id-operation-param-name="operation.id">` В ситуации, когда в параметрах точки включен параметр **«Получение ID...»**, запрос резервирования отправится по умолчанию, т. е. дополнительно задавать в сценарии его не нужно (действует для первого advanced-запроса). В этом случае номер ID сохранится по умолчанию под ключом **operation-id**	
id-operation-param-name	В значении указывается ключ, в котором сервер вернет на терминал значение зарезервированного ID. Идентификатор будет доступен: 1. В сценарии по заданному ключу. 2. Для вывода в чеке при помощи **$operation.serverIdOperation** (подробнее см. в документе «Шаблон чека»). **Пример:** `id-operation-param-name="id2"`	Нет
hidden-execute	Может принимать значения **true**, **false**. При заданном значении **true** будет скрыто окно ожидания выполнения действия при исполнении онлайн-запроса. Обрабатывается ТПО 5.	Нет

Действие `<online-request>` представляет собой активную часть экрана в экране, поэтому он также использует элемент `<actions>` (элемента `<fields>` в нем нет), но в отличие от кнопок `<action>` элемента `<dialog>` действия `<action>` в онлайн-запросе определяют реакцию на результат выполнения этого запроса. Тип ответа указывается в атрибуте **type** элемента `<action>`.

Результатом может быть:

1. **Успех (success)** — означает успешное выполнение запроса: на сервере запрос выполнился как ожидалось, и вернул в контекст те результаты, которые предполагаются согласно его спецификации.

2. **Ошибка (error)** — означает, что во время выполнения на сервере провайдера возникла исключительная ситуация. Как правило, это означает потерю сетевого соединения между сервером процессинга и провайдером, либо ответ провайдер не по протоколу (не удалось разобрать ответ, извлечь нужные данные). Как правило, в таком случае запрос не возвращает никаких атрибутов в контекст сценария. В таком случае, возможно в сценарии обработать эту ситуацию и повторить запрос/вернуться на предыдущий шаг. Так же можно перехватить ошибки, возвращаемые шлюзом, и переопределить их, например:

```
<action type="error">
 <if condition="service_error == 2">
  <then>
   <dialog type="Error" title="Внимание!" message="Аккаунт заблокирован!"
      timeout="10" default="okay">
    <actions>
     <action type="okay" title="OK">
      <cancel/>
     </action>
    </actions>
   </dialog>
  </then>
 </if>
</action>
```

Если необходимо вывести на экран текст ошибки, которая вернулась в результате онлайн-запроса, то в блок обработки ошибки необходимо добавить вывод переменных **`service_error`**, **`serverError`**. Имена переменных предопределены.

Пример:

```
<dialog type="Error" title="Error" message="The UserName does not exist.
{0} {1}" mess-params="service_error,serverError" timeout="10"
default="okay"
```

В примере в **mess-params** указано, что необходимо вывести значение переменных **service_error**, **serverError**, в **title** осуществлен вывод значений переменных посредством использования конструкции {номер переменной}. Нумерация переменных начинается с 0.

3. Исключение (exception) — возникает при невозможности отправить запрос на сервер с терминала или при неполучении ответа терминалом от сервера. Отсутствие связи между терминалом и ПЦ. Не возвращается никаких атрибутов в контекст сценария, как правило, в таком случае работу сценария надо завершить.

Стандартная конфигурация секции `<actions>` в онлайн-запросе:

```
<actions>
   <action type="success"> ... </action>
   <action type="error"> ... </action>
   <action type="exception"> ... </action>
</actions>
```

В случае, когда онлайн-запрос и проведение платежа проходят по разным провайдерам, необходимо сделать соответствующие настройки в кабинете. Для этого в разделе «Диспетчерская — Направление» проведения платежа следует выставить для сервиса провайдеров для проведения платежа и онлайн-запроса. По умолчанию используется один и тот же провайдер.

Пример:

```
<online-request type="default" function="find-departure-point"
params="departure-point">
 <actions>
  <goto-action type="success" target="2screen"/>
  <action type="error">
   <dialog type="Info" title="Ошибка"
           message="Невозможно выполнить запрос" timeout="10"
           default="okay">
    <actions>
     <action type="okay" title="OK">
      <cancel/>
     </action>
    </actions>
   </dialog>
  </action>
```

```
</actions>
</online-request>
```

Данный онлайн-запрос выполняется на сервере при помощи функции **find-departure-point**, которой передается переменная **departure-point**. При ответе сервера **«success»** (успех) переходим на следующий экран (2screen). При ответе сервера **«error»** (ошибка) открывается диалоговое окно: «Невозможно выполнить запрос». Подробно диалоговые окна описаны в разделе 6.5.

Данные, которые были возвращены в результате запроса, выводятся на экране 2screen:

```
<screen type="Selector" decor="list" id="2screen"
        title="Пункт отправления">
 <fields>
   <selector-field id="departure-point" title="Пункт отправления">
     <items type="data" key="departure-point-list"/>
   </selector-field>
 </fields>
 <actions>
     ...
 </actions>
</screen>
```

В атрибуте **key** элемента <items> указывается название переменной, возвращаемой в результате онлайн-запроса. Кроме того, в случае успешного ответа в результате онлайн-запроса, полученные данные возможно передать на экран подтверждения.

Пример:

```
<screen ...>
 ...
 <actions>
  <action type="Next" title="ДАЛЕЕ">
   <online-request type="default" function="" params="id1">
    <actions>
     <goto-action type="success" target="confirm"/>
     ...
    </actions>
   </online-request>
  </action>
  ...
 </actions>
```

```
 </screen>
 <!--Создание экрана подтверждения-->
 <screen type="Confirm" decor="simple" title="Информация" id="confirm">
  <!--Список атрибутов, отображаемых на экране подтверждения-->
  <fields list="id1"/>
  <actions>
   <!--Переход на экран оплаты-->
   <goto-action type="Next" target="pay"/>
   ...
  </actions>
 </screen>
```

По ключу возвращаемых переменных возможно выполнить какие-либо операции над ними.

Пример:

```
<online-request type="default" function="get-order"
   params="trip-id,departure-point,destination-point,trip-ticket-kind">
  <actions>
   <action type="success">
    <modify src-key="trip-seat-num" t-key="^(.*)$" v-key="id1"
           t-value="^(.*$)" v-value="Место $1" t-value-title="^(.*$)"
           v-value-title="Место $1"/>
    <modify src-key="trip-id" t-key-title="^(.*)$"
            v-key-title="ID поезда"/>
    <modify src-key="account-id" t-key-title="^(.*)$"
           v-key-title="Номер брони"/>
    <modify src-key="trip-seat-num" t-key-title="^(.*)$"
           v-key-title="Место"/>
    <modify src-key="trip-platform-number" t-key-title="^(.*)$"
            v-key-title="Платформа"/>
    <goto target="confirm"/>
   </action>
  </actions>
</online-request>
```

6.7 ГЕНЕРАЦИЯ ДВУМЕРНОГО ШТРИХКОДА

Для возможностей генерации двумерного штрихкода в сценарии используется действие `<barcode>`.

Таблица 6.7.1 — Атрибуты действия `<barcode>`.

Атрибут	Описание	Обяз.
text	Сообщение, которое будет зашифровано в штрихкоде. В сообщении возможно использовать параметры, например **text="left ${id2} right"**.	Да
type	Тип генерируемого штрихкода. На данный момент доступно использование только типа **«qr»**.	Да
width/height	Атрибуты, позволяющие задать размеры штрихкода по ширине и высоте соответственно.	Да
target-file-name	Путь и имя файла, в который будет сохранено изображение сгенерированного штрихкода. Указывается относительно корня хранилища ТПО, в формате *.png	Да

Пример:

```
<barcode text="loren ipsum" type="qr" width="200" height="200" target-file-name="test.png"/>
```

6.8 СЛОЖНАЯ ВАЛИДАЦИЯ ДАННЫХ (<COMPLEX-VALIDATOR>)

6.8.1 ОБЩИЕ СВЕДЕНИЯ

Проверка введенных данных на соответствие различным условиям осуществляется с использованием действия `<complex-validator>`.

Дочерние элементы: `<actions>`.

Атрибуты зависят от типа валидатора, задаваемого атрибутом **`type`**.

6.8.2 ПРОВЕРКА НА СООТВЕТСТВИЕ БИКА

Валидатор `<complex-validator>` с типом **`type=«bank-account»`** осуществляет проверку на соответствие БИКа.

Атрибуты:

1. **`bik-key`** (для ТПО версий 5, 7) — ключ атрибута с БИКом банка.

2. **`bank-key`** (для ТПО версий 5, 7) — ключ атрибута, в который в случае успеха будет записан результат. Если ключ не указан, то результат будет записан под ключом «bank». Названия банков берутся из файла *`<корневой каталог ТПО>/resources/bik.csv`* в 5 версии ТПО и файла *`<корневой каталог ТПО>/res/bik.csv`* в 7 версии ТПО.

3. **account-key** (для ТПО версий 5, 7) — ключ атрибута с номером расчетного счета банка.

Для данного типа валидатора доступны следующие типы `<action>`:

1. `<action type="bik-error"> ... </action>` — действие в случае, если в файле *bik.csv* не найден банк, соответствующий введенному БИК.

2. `<action type="account-error"> ... </action>` — действие в случае, если номер расчетного счета банка не соответствует БИК.

3. `<action type="success"> ... </action>` — действие в случае успешного прохождения проверок.

Файл *bik.csv* содержит строки по 3 поля, разделённых символом";":

1. **0** — не используется, то есть поле может быть указано, но обработано не будет.

2. **1** — название банка.

3. **2** — БИК.

Фрагмент файла bik.csv

```
;ПУ БАНКА РОССИИ N 03148;040012002
;ПУ БАНКА РОССИИ N 83604;040021002
;ПУ БАНКА РОССИИ N 83524;040031002
БАЙКОНУР;ПУ БАНКА РОССИИ N 25631;040037002
;ПУ БАНКА РОССИИ N 10513;040041002
;ПУ БАНКА РОССИИ N 64106;040047002
БИЙСК;РКЦ БИЙСК;040147000
КОСТРОМА;ОАО КБ "РЕГИОНАЛЬНЫЙ КРЕДИТ";043469751
БИЙСК;ОАО "НАРОДНЫЙ ЗЕМЕЛЬНО-ПРОМЫШЛЕННЫЙ БАНК";040147781
БАРНАУЛ;ООО КБ "АЛТАЙКАПИТАЛБАНК";040173771
РУБЦОВСК;РКЦ РУБЦОВСК;040155000
БАРНАУЛ;АКБ "АЛТАЙБИЗНЕС-БАНК" (ОАО);040173725
БАРНАУЛ;ООО "КБ "ТАЛЬМЕНКА-БАНК";040173749
БАРНАУЛ;ОТДЕЛЕНИЕ БАРНАУЛ;040173001
БАРНАУЛ;АЛТАЙСКОЕ ОТДЕЛЕНИЕ N8644 ПАО СБЕРБАНК;040173604
БАРНАУЛ;БАРНАУЛЬСКИЙ Ф БАНКА"ВОЗРОЖДЕНИЕ"(ПАО);040173735
```

```
БАРНАУЛ;"СИБСОЦБАНК" ООО;040173745
СИБИРСКИЙ;ПУ БАНКА РОССИИ ШОЛОХОВСКОЕ;040181002
АНАПА;РКЦ АНАПА;040304000
АРМАВИР;РКЦ АРМАВИР;040306000
ГОРЯЧИЙ КЛЮЧ;РКЦ ГОРЯЧИЙ КЛЮЧ;040314000
РОСТОВ-НА-ДОНУ;ООО КБ "РОСТФИНАНС";046027283
КРОПОТКИН;РКЦ КРОПОТКИН;040326000
КРАСНОДАР;АКБ "КРЫЛОВСКИЙ" (ОАО);040349569
КРАСНОДАР;ЮЖНОЕ ГУ БАНКА РОССИИ;040349001
КРАСНОДАР;КРАСНОДАРСКОЕ ОТДЕЛЕНИЕ N8619 ПАО СБЕРБАНК;040349602
КРАСНОДАР;БАНК "ПЕРВОМАЙСКИЙ" (ПАО);040349715
КРАСНОДАР;АО "КУБАНЬТОРГБАНК";040349718
КРАСНОДАР;КБ "КУБАНЬ КРЕДИТ" ООО;040349722
КРАСНОДАР;КБ "КУБАНСКИЙ УНИВЕРСАЛЬНЫЙ БАНК" (ООО);040349745
КРАСНОДАР;ПАО "ИДЕЯ БАНК";040349772
КРАСНОДАР;Ф-Л БАНКА ГПБ (АО) В Г. КРАСНОДАРЕ;040349781
КРАСНОДАР;ООО КБ "НОВОПОКРОВСКИЙ";040349808
```

Пример:

```
<!--Проверка на соответствие БИК-->
<complex-validator type="bank-account" bik_key="bik" account_key="id1">
 <actions>
  <!--Описание поведения сценария оплаты в случае успешной проверки-->
  <action type="success">
   <goto target="2screen"/>
  </action>
  <!--Описание поведения сценария оплаты в случае ошибки проверки БИКа-->
  <action type="bik_error">
   <!--Отображение диалога с сообщением об ошибке-->
   <dialog type="Error" title="Ошибка"
           message="Не найден банк соответствующий
           введенному БИК" timeout="10" default="okay">
    <actions>
     <!--Описание поведения сценария оплаты при нажатии на диалоговом
окне кнопки "ОК"-->
     <action type="okay" title="ОК"/>
    </actions>
   </dialog>
   <cancel/>
  </action>
  <!--Описание поведения сценария оплаты в случае ошибки проверки номера
счета-->
```

```
  <action type="account_error">
   <!--Отображение диалога с сообщением об ошибке-->
   <dialog type="Error" title="Ошибка"
           message="Номер счета не соответствует БИК" timeout="10"
           default="okay">
    <actions>
     <!--Описание поведения сценария оплаты при нажатии на диалоговом
окне кнопки "ОК"-->
     <action type="okay" title="ОК"/>
    </actions>
   </dialog>
   <cancel/>
  </action>
 </actions>
</complex-validator>
```

6.8.3 ПРОВЕРКА ПО НОМЕРУ ЛИЦЕВОГО СЧЕТА

Валидатор `<complex-validator>` с типом **`type=«local-capacity»`** осуществляет проверку по номеру лицевого счета. В 7 версии ТПО не используется.

Атрибуты:

1. **`account-key`** — номер лицевого счета.

2. **`function`** — выражение, на соответствие которому осуществляется проверка.

Пример:

```
<!--Проверка по номеру лицевого счета-->
<complex-validator type="local-capacity" account-key="id3"
function="capacity">
 <actions>
  <!--Описание поведения сценария оплаты в случае успешной проверки-->
  <action type="success">
   <goto target="pay"/>
  </action>
  <!--Описание поведения сценария оплаты в случае ошибки проверки-->
```

```
  <action type="error">
   <!--Отображение диалога с сообщением об ошибке-->
   <dialog type="Error" title="Ошибка" message="Не найден номер"
           timeout="10" default="okay">
    <actions>
     <!--Описание поведения сценария оплаты при нажатии на диалоговом
окне кнопки "ОК"-->
     <action type="okay" title="ОК">
      <cancel/>
     </action>
    </actions>
   </dialog>
   <cancel/>
  </action>
 </actions>
</complex-validator>
```

6.8.4 ПРОВЕРКА ПО НОМЕРНЫМ ЕМКОСТЯМ

Валидатор `<complex-validator>` с типом **`type=«capacity»`** осуществляет проверку по номерным емкостям.

Атрибуты:

1. **`number`** — ключ атрибута с номером, который необходимо проверить;

Пример:

```
<!--Проверка по номерным ёмкостям-->
<complex-validator type="capacity" number="chnum">
 <actions>
  <action type="success">
   <!--Выполнение онлайн запроса-->
   <online-request type="default" function="getticketprice"
                   params="trip-id,departure-point,destination-
                   point,trip-ticket-kind" attempts="5" timeout="1"
                   timeoutall="60">
    <actions>
```

```
      <!--Если в результате онлайн-запроса вернулся успех-->
      <action type="success">
      <!--Передача значения числовой переменной в сумму платежа без учета
комиссии-->
       <set-sum from="trip-ticket-price" type="units"/>
       <!--Модификация параметра trip-ticket-price-->
       <modify src-key="trip-ticket-price" t-key-title="^(.*)$"
              v-key-title="Цена билета"/>
       <!--Модификация параметра phone-->
       <modify src-key="phone" t-key="^(.*)$" v-key="id1"
              t-value="^(.*$)" v-value="$1" t-value-title="^(.*$)"
              v-value-title="$1" v-key-title="Номер телефона"/>
       <goto target="confirm1"/>
      </action>
      <!--Если в результате онлайн-запроса вернулась ошибка-->
      <action type="error">
       <!--Отображение диалога с сообщением об ошибке-->
       <dialog type="Info" title="Ошибка" message="Невозможно получить
              данные о цене билета. Повторите попытку."
              timeout="1" default="okay">
        <actions>
         <!--Описание поведения сценария, если нажата кнопка "OK"-->
         <action type="okay" title="OK">
          <cancel/>
         </action>
        </actions>
       </dialog>
      </action>
      <action type="exception">
       <!--Отображение диалога с сообщением об ошибке-->
       <dialog type="Info" title="Ошибка" message="Невозможно связаться с
              сервером. Повторите попытку." timeout="1" default="okay">
        <actions>
         <!--Описание поведения сценария, если нажата кнопка "OK"-->
         <action type="okay" title="OK">
          <cancel/>
         </action>
        </actions>
       </dialog>
      </action>
     </actions>
    </online-request>
   </action>
```

```
 <!--Описание поведения сценария оплаты в случае ошибки проверки-->
 <action type="error">
  <!--Отображение диалога с сообщением об ошибке-->
  <dialog type="Error" title="Error" message="Ошибка проверки номера"
          timeout="1" default="okay">
   <actions>
    <!--Описание поведения сценария, если нажата кнопка "Продолжить"-->
    <action type="okay" title="Продолжить">
     <!--Модификация параметра phone-->
     <modify src-key="phone" t-key="^(.*)$" v-key="id1" t-value="^(.*$)"
             v-value="$1" t-value-title="^(.*$)" v-value-title="$1"
             v-key-title="Номер телефона"/>
     <!--Модификация параметра trip-ticket-price-->
     <modify src-key="trip-ticket-price" t-key-title="^(.*)$"
             v-key-title="Цена билета"/>
     <!--Удаление переменных из контекста-->
     <clear keys="chnum"/>
     <goto target="confirm1"/>
    </action>
    <!--Описание поведения сценария, если нажата кнопка "Вернуться"-->
    <action type="no" title="Вернуться">
     <goto target="phone_screen1"/>
    </action>
   </actions>
  </dialog>
 </action>
</actions>
</complex-validator>
```

6.8.5 ПРОВЕРКА НА ЗАПРЕЩЕННЫЕ ЗНАЧЕНИЯ

Валидатор `<complex-validator>` с типом **type=«block»** предназначен для выполнения проверки в соответствии с файлом, указанным в **function**. **Атрибуты**:

1. **key** — ключ атрибута, проверка которого осуществляется.

2. **function** — наименование файла, по которому будет осуществляться проверка. Например, function=«currency». В этом случае значение атрибута, указанного в **key**,

будет сравниваться со значениями, указанными в файле *<значение function>.data*. И в случае совпадения со значениями в файле, возникнет ошибка. Размещаться файл в 5 и 7 версии ТПО должен в каталоге ***<корень ТПО>/resources/block-data/***. Расширение файла всегда .data.

Пример (для 5 и 7 версии ТПО):

```
<scenario xmlns="http://pay-logic.ru" begin="0screen">
 <!-- E-Finance  -->
 <!--Создание экрана ввода данных с несколькими полями-->
 <screen type="group" title="Тип оплаты" id="0screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 11. Название поля
ввода: "№ договора"-->
   <text-field id="id1" title="№ договора" max-len="11"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules> <rule regex="^\d{11}$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого номера на
экране-->
    <formatter type="regex">
     <rules default="** *** *** ***"/>
    </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите № договора </help>
   </text-field>
   <text-field id="fio" title="ФИО плательщика" max-len="60"
               keyboard="any:[ru,symb]:upper:true">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
    <rules>
     <rule regex="^\D{1,10}$"/>
    </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> Введите Ваши ФИО </help>
   </text-field>
  </fields>
  <actions>
```

```
  <!--Описание поведения сценария оплаты при нажатии "Далее"-->
  <action type="Next" title="ДАЛЕЕ">
   <!--Проверка id1 на запрещенные значения-->
   <complex-validator type="block" key="id1" function="currency">
    <actions>
     <!--Если в результате проверки вернулся успех-->
     <action type="success">
      <goto target="pay"/>
     </action>
     <!--Если в результате проверки вернулась ошибка-→
     <action type="error">
      <!--Отображение диалога с сообщением об ошибке-->
      <dialog type="Error" title="Ошибка"  message="Wrong data"
             timeout="10" default="okay">
       <actions>
        <!--Описание поведения сценария, если нажата кнопка "OK"-->
        <action type="okay" title="OK"/>
       </actions>
      </dialog>
      <cancel/>
     </action>
    </actions>
   </complex-validator>
  </action>
  <!--Описание поведения сценария оплаты при нажатии "Назад"-->
  <action type="Prev" title="Назад">
   <goto target="previous"/>
  </action>
  <!--Описание поведения сценария оплаты при нажатии "Выход"-->
  <action type="Exit" title="ВЫХОД">
   <goto target="exit"/>
  </action>
 </actions>
</screen>
</scenario>
```

Пример содержимого currency.data:

```
11111111111
22222222222
33333333333
```

При вводе любого из значений, указанных в файле currency.data в качестве № договора (**id1**) в примере, пользователю появится диалоговое окно с надписью «Wrong data».

6.8.6 ПРОВЕРКА С УЧЕТОМ МАКСИМАЛЬНОЙ СУММЫ ПО СЕРВИСУ

Валидатор `<complex-validator>` с типом **type=«purchase-validator»** проверяет введенную сумму с учетом максимальной суммы по сервису и комиссии, то есть проверяет, что заданная сумма вместе с суммой комиссией не будет превышать максимальное значение платежа по сервису. Доступен в версиях ТПО 5 (7im), 7.

Атрибуты:

1. **sum**— хранит значение проверяемой суммы. Обязательный атрибут. Непустая строка до 25 символов.

2. **check-min** — определяет необходимость проверки на минимум. Возможные значения:

   1) **true** — проверка необходима;

   2) **false** — проверка не требуется.

Пример:

```
<!--Проверка введённой суммы-->
<complex-validator type="purchase-validator" sum="ssum1">
 <actions>
  <!--Описание поведения сценария оплаты в случае успешной проверки-->
  <action type="success">
   <!--Если #flag не равно null-->
   <if condition="is-not-null #flag">
    <then>
     <!--Удаление переменных из контекста-->
     <clear keys="#flag"/>
     <!--Переход на экран с id meters_screen-->
     <goto target="meters_screen"/>
```

```
    </then>
    <else>
     <Иначе: если account-id эквивалентно ^\d{1,25}$>
     <if condition="account-id ~ ^\d{1,25}$">
      <then>
       <!--Отправка онлайн-запроса-→
       <online-request type="default" function="get-meters-ls"
                       params="id1,town-id,account-id" attempts="5"
                       timeout="10" timeout-all="60">
        <actions>
         <!--Описание поведения сценария, если в ответ на онлайн-запрос
вернулся успех-->
         <action type="success">
          <!--Удаление переменных из контекста-->
          <clear keys="id1"/>
          <!--Переход на экран с id meters_screen-->
          <goto target="meters_screen"/>
         </action>
         <!--Описание поведения сценария, если в ответ на онлайн-запрос
вернулась ошибка-->
         <action type="error">
          <!--Отображение диалога с сообщением об ошибке-->
          <dialog type="Error" title="Информация" message="Невозможно получить
                  список счетчиков. После нажатия на кнопку ОК будет совершен
                  переход на экран подтверждения" timeout="1" default="okay">
           <actions>
            <!--Описание поведения сценария, если на диалоге нажата кнопка
"OK"-->
            <action type="okay" title="OK">
             <goto target="request_error"/>
            </action>
           </actions>
          </dialog>
         </action>
         <!--Описание поведения сценария, если в ответ на онлайн-запрос
вернулось исключение-→
         <action type="exception">
          <!--Отображение диалога с сообщением об ошибке-→
          <dialog type="Error" title="Error" message="Невозможно получить
                  список счетчиков в данный момент." timeout="1"
                 default="okay">
           <actions>
```

```
            <!--Описание поведения сценария, если на диалоге нажата кнопка
"OK"-->
            <action type="okay" title="OK">
             <cancel/>
            </action>
           </actions>
          </dialog>
         </action>
        </actions>
       </online-request>
      </then>
      <else>
       <!--Отправка онлайн-запроса-->
       <online-request type="default" function="get-meters-universal"
                       params="id1,town-id,street-id,building-
                       number,appartment-number" attempts="5" timeout="10"
                       timeout-all="60">
        <actions>
         <!--Описание поведения сценария, если в ответ на онлайн-запрос
вернулся "Успех"-->
         <action type="success">
          <!--Удаление переменных из контекста-->
          <clear keys="id1"/>
          <!--Переход на экран с id "meters_screen"-->
          <goto target="meters_screen"/>
         </action>
         <!--Описание поведения сценария, если в ответ на онлайн-запрос
вернулась "Ошибка"-->
         <action type="error">
          <!--Отображение диалога с сообщением об ошибке-->
          <dialog type="Error" title="Информация" message="Невозможно получить
                  список счетчиков. После нажатия на кнопку ОК будет совершен
                  переход на экран подтверждения" timeout="1" default="okay">
           <actions>
            <!--Описание поведения сценария, если нажата кнопка "OK"-->
            <action type="okay" title="OK">
             <goto target="request_error"/>
            </action>
           </actions>
          </dialog>
         </action>
         <!--Описание поведения сценария, если в ответ на онлайн-запрос
вернулось исключение-->
```

```
        <action type="exception">
         <!--Отображение диалога с сообщением об ошибке-->
         <dialog type="Error" title="Error" message="Невозможно получить
              список счетчиков в данный момент." timeout="1" default="okay">
          <actions>
           <!--Описание поведения сценария, если нажата кнопка "OK"-->
           <action type="okay" title="OK">
            <cancel/>
           </action>
          </actions>
         </dialog>
        </action>
       </actions>
      </online-request>
     </else>
    </if>
   </else>
  </if>
 </action>
 <action type="error">
  <!--Отображение диалога с сообщением об ошибке-->
  <dialog type="Error" title="Error" message="Общая сумма платежа превысила
         {0}. Скорректируйте суммы услуг." mess-params="max_purchase"
         timeout="10" default="okay">
   <actions>
    <!--Описание поведения сценария, если нажата кнопка "OK"-->
    <action type="okay" title="OK">
     <goto target="services_screen"/>
    </action>
   </actions>
  </dialog>
 </action>
</actions>
</complex-validator>
```

6.9 ЗАПРОС К ОБОРУДОВАНИЮ (<HDW-REQUEST>)

Для запроса к оборудованию (смарт-карте, кард-ридеру, диспенсеру карт) используется действие `<hdw-request>`. Структура действия:

```
<hdw-request type=[smart-card|card-reader|card-dispenser]
             function=<notEmptyString15> params=<keyListType>>
      <actions>
          <action type=[success|error|exception]>
                ...
          </action>
      </actions>
</hdw-request>
```

Дочерние элементы: `<actions>`.

Атрибут `params` аналогичен атрибуту для `<online-request>` (раздел 6.6), остальные атрибуты приведены в таблице 6.9.1.

Таблица 6.9.1 — Атрибуты действия `<hdw-request>`

Атрибут	Описание	Обяз.
`type`	Тип выполняемого запроса. Для обеих версий ТПО доступны: • **`smart-card`** — запрос к смарт-карте; • **`card-reader`** — запрос к кард-ридеру. Только для ТПО 7: • **`card-dispenser`** — запрос к диспенсеру карт; • **`webcam`** — запрос к веб-камере; • **`doc-parser`** — запрос на распознавание изображения документа; • **`doc-scanner`** — запрос к сканеру документов;	Да

Атрибут	Описание	Обяз.
	• **locker** — запрос к контроллеру замков; • **relay** — запрос к контроллеру контактов; • **cash-dispenser** — запрос к устройствам выдачи денег; • **send-photo** — запрос отправки файлов (фотографий); • **printer** — запрос информации о ККМ. Любой запрос вернет **success** в случае успеха и **exception** — в случае ошибки. Подробнее каждый из запросов описан в последующих разделах	
function	Название функции, выполняемой на стороне сервера процессинга при выполнении запроса к оборудованию. При запросе к смарт-карте функции определяются реализацией scard-handler'a, который указывается в рутовом теге сценария	Да
params	Передаваемые переменные при запросе, по которым необходимо вернуть информацию, перечисляются через запятую без пробелов. Должен быть указан хотя бы один параметр. Возможно также указание всех имеющихся переменных без перечисления через $all Если провайдер возвращает комиссию поставщика и в шлюзе реализована поддержка запроса комиссии, то в онлайн-запросе с функцией `get-fee` следует передавать параметры «**amount-without-fee,fee-calculation-method**»	Да
hdw-params	Используется при запросе к веб-камере в случае, когда на ТПО настроено несколько камер. В атрибуте возможно указать, какую камеру необходимо использовать. Например: `<hdw-request type="webcam" function="take-`	Нет

Атрибут	Описание	Обяз.
	`photo" params="save-file" hdw-params="2">` В примере в **hdw-params** указано значение **«2»**, это означает, что использоваться будет вторая камера (максимальное количество — 3). Если в сценарии оплаты отсутствует данный атрибут, по умолчанию будет работать первая камера.	

6.9.1 ЗАПРОС К КАРД-РИДЕРУ (CARD-READER)

При запросе к кард-ридеру доступны следующие функции:

1. `enable` — включить прием карт.
2. `eject` — вернуть карту.
3. `disable` — выключить прием карт.
4. `release` — отключить ридер.
5. `read-track1` — прочитать магнитный трек №1. Вернет InputElement с id 'track1'.
6. `read-track2` — прочитать магнитный трек №2. Вернет InputElement с id 'track2'.
7. `read-track3` — прочитать магнитный трек №3. Вернет InputElement с id 'track3'.

В примере ниже при успешном включении приема карт кард-ридером происходит переход на экран **`"insert_screen"`**, в случае ошибки — отображается соответствующее диалоговое окно.

Пример:

```
<hdw-request type="card-reader" function="enable" params="$all">
  <actions>
    <!--Описание поведения сценария, если на запрос возвращен "Успех"-->
    <goto-action type="success" target="insert_screen"/>
    <!--Описание поведения сценария, если в результате запроса возникло
исключение-->
    <action type="exception">
      <!--Отображение диалога с сообщением об ошибке-->
      <dialog type="Info" title="Ошибка" message="Ошибка кардридера"
              timeout="10" default="okay">
        <actions>
```

```
            <!--Описание поведения сценария, если нажата кнопка "OK"-->
            <action type="okay" title="OK">
              <cancel/>
            </action>
          </actions>
        </dialog>
      </action>
    </actions>
</hdw-request>
```

6.9.2 ЗАПРОС К ДИСПЕНСЕРУ КАРТ (CARD-DISPENSER)

При запросе к диспенсеру карт доступны следующие функции:

1. `dispense` — производит попытку выдать карту. Если выдача прошла успешно, то в переменной **`is-dispense`** будет значение **`true`**, иначе **`false`**. При выполнении желательна проверка возвращаемого значения.

2. `bc-dispense` — после завершения данной функции в контексте оказывается переменная **`key=#cvd-barcode`** со значением считанного штрих-кода. В качестве параметра функции (атрибут **`params`**) передается переменная с регулярным выражением проверки штрих-кода.

В следующем примере первоначально определяется регулярное выражение (переменная #bc-regex), в соответствии с которым необходимо вернуть информацию в результате выполнения запроса к диспенсеру карт. Затем выполняется запрос (<hdw-request type="card-dispenser" ...>) к диспенсеру карт для считывания шрих-кода. Шрих-код распознается в соответствии с регулярным выражением, определенным в переменной #bc-regex, и сохраняется в переменную #cvd-barcode, которая затем выводится на экране подтверждения.

Пример:

```
<scenario begin="pay_type" >
 <!--Создание экрана выбора-->
 <screen type="Selector" decor="button" title="Поиск штрафов"
         id="pay_type">
  <fields>
   <!--Создание поля выбора типа платежа-->
   <selector-field id="type" title="Тип платежа">
    <items type="static">
     <item title="Выдать карту" value="1"/>
     <item title="Выдать карту + штрих код" value="2"/>
    </items>
   </selector-field>
  </fields>
  <actions>
   <!--Описание поведения сценария, если нажата кнопка "Next"-->
   <action type="Next" title="NEXT">
    <!--Если type равно 1-->
    <if condition="type == 1">
     <then>
      <!--Запрос к диспенсеру карт-->
      <hdw-request type="card-dispenser" function="dispense">
       <actions>
        <!--Описание поведения сценария, если запрос завершился
успехом-->
        <action type="success">
         <!--Отображение диалога с сообщением об ошибке-->
         <dialog type="Info" title="Success" message="{0}"
                 timeout="1" default="okay" mess-params="i">
          <actions>
           <!--Описание поведения сценария, если нажата кнопка
"Continue"-->
           <action type="okay" title="Continue">
            <goto target="exit"/>
           </action>
          </actions>
         </dialog>
        </action>
        <!--Описание поведения сценария, если запрос завершился
ошибкой-->
        <action type="error">
         <!--Отображение диалога с сообщением об ошибке-->
         <dialog type="Error" title="Dispenser Error"
```

```
                    message="Выдача карт временно недоступна"
                    timeout="10" default="okay">
              <actions>
                <!--Описание поведения сценария, если нажата кнопка
"Continue"-->
                <action type="okay" title="Continue">
                  <goto target="exit"/>
                </action>
              </actions>
            </dialog>
          </action>
          <!--Описание поведения сценария, если запрос завершился
исключением-->
          <action type="exception">
            <!--Отображение диалога с сообщением об ошибке-->
            <dialog type="Error" title="Dispenser exception"
                    message="Диспенсер карт неисправен " timeout="10"
                    default="okay">
              <actions>
                <!--Описание поведения сценария, если нажата кнопка
"Continue"-->
                <action type="okay" title="Continue">
                  <goto target="exit"/>
                </action>
              </actions>
            </dialog>
          </action>
        </actions>
      </hdw-request>
    </then>
    <else>
      <!--Иначе: объявление переменной #bc-regex-->
      <set key="#bc-regex" value=".*2" />
      <!--Запрос к диспенсеру карт-->
      <hdw-request type="card-dispenser" function="bc-dispense"
                   params="#bc-regex">
        <actions>
          <!--Описание поведения сценария, если запрос завершился
успехом-->
          <action type="success">
            <!--Отображение диалога с информационным сообщением-->
            <dialog type="Info" title="Success" message="{0}"
                    timeout="1" default="okay" mess-params="i">
```

```
              <actions>
                <!--Описание поведения сценария, если нажата кнопка
"Continue"-->
                <action type="okay" title="Continue">
                  <goto target="confirm"/>
                </action>
              </actions>
            </dialog>
          </action>
          <!--Описание поведения сценария, если запрос завершился
ошибкой-->
          <action type="error">
            <!--Отображение диалога с сообщением об ошибке-->
            <dialog type="Error" title="Dispenser Error"
                    message="Выдача карт временно недоступна"
                    timeout="10" default="okay">
              <actions>
                <!--Описание поведения сценария, если нажата кнопка
"Continue"-->
                <action type="okay" title="Continue">
                  <goto target="exit"/>
                </action>
              </actions>
            </dialog>
          </action>
          <!--Описание поведения сценария, если запрос завершился
исключением-→
          <action type="exception">
            <!--Отображение диалога с сообщением об ошибке-->
            <dialog type="Error" title="Dispenser exception"
                    message="Диспенсер карт неисправен" timeout="10"
                    default="okay">
              <actions>
                <!--Описание поведения сценария, если нажата кнопка
"Continue"-->
                <action type="okay" title="Continue">
                  <goto target="exit"/>
                </action>
              </actions>
            </dialog>
          </action>
        </actions>
      </hdw-request>
```

```
      </else>
     </if>
    </action>
    <action type="Prev" title="Назад">
     <goto target="exit"/>
    </action>
    <action type="Exit" title="Выход">
     <goto target="exit"/>
    </action>
  </actions>
 </screen>
 <!--Создание экрана подтверждения данных-->
 <screen type="Confirm" decor="simple" title="Проверка данных"
         id="confirm">
  <!--Список атрибутов, отображаемых на экране подтверждения-->
  <fields list="#cvd-barcode"/>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии "Далее"-->
   <action type="Next" title="ДАЛЕЕ">
    <goto target="exit"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии "Назад"-->
   <action type="Prev" title="НАЗАД">
    <goto target="exit"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии "Выход"-->
   <action type="Exit" title="ВЫХОД">
    <goto target="exit"/>
   </action>
  </actions>
 </screen>
</scenario>
```

6.9.3 ЗАПРОС НА РАСПОЗНАВАНИЕ ИЗОБРАЖЕНИЯ ДОКУМЕНТА (DOC-PARSER)

Запрос осуществляется через переменную save-file, в значении которой указывается путь до нужного изображения.

Пример:

```
<set key="save-file" value="<путь до изображения>”/>
<!--Запрос на распознавание -->
<hdw-request type="doc-parser" function="parse" params="save-file">
```

При запросе используется функция `parse`, которая распознает изображение документа (например, страницу паспорта).

Для успешного распознавания:

1. В папку ***/res/doc/*** разместите файл с базой для библиотеки и наименованием bundle.zip. Пример: ***/res/doc/bundle.zip***.

2. В сборку в директорию ***/jni/<каталог с нужной разрядностью>*** добавьте библиотеку libjniSmartIdEngine. Пример для Java x64: ***/jni/64/libjniSmartIdEngine.so.***

3. Дополнительно в сборку, в директорию ***/lib***, добавьте файл jniSmartIdEngineJar.jar

В ответ вернутся распознанные атрибуты, помещенные в контекст сценария. Например, значения следующих и других ключей:

```
"name"
"last_name"
"surname"
"patronymic"
"birth_date"
"birthdate"
"expiry_date"
"issue_date"
"address"
```

6.9.4 ЗАПРОС К СКАНЕРУ ДОКУМЕНТОВ (DOC-SCANNER)

Пример запроса:

```
<hdw-request type="doc-scanner" function="take-image" params="save-file">
```

Доступны следующие функции:

1. `take-image` — получить изображения.
2. `take-data` — получить данные.
3. `take-all` — получить изображение и данные.

Таблица 6.9.4.1 — Параметры функции **`take-image`**

Параметры	Описание	Обяз.
`save-file`	Путь для сохранения изображения. Если не указан, то изображение не будет сохранено	Нет
`image-key`	Ключ, под которым в контекст будет помещен инпут-элемент с изображением в формате base64	Нет

При использовании функции `take-data` распознанные данные помещаются в контекст. Ключи атрибутов берутся из данных от сканера.

Примеры ключей и значений атрибутов:

```
Sex =M
Birthday =911225
PersonalNo =
Checksum =true
MRTDs
=P<IVANOV<<IVAN<<<<<<<<<<<<<<<<<<<<1111111111RUS9112251M1111111<<<<<<<<
<<<<
DocumentNo =111111111
IssueState =RUS
Nationality =RUS
```

```
Type =P
Angle =0
Familyname =IVANOV
NativeName =
Givenname =IVAN
Dateofexpiry =270610
```

Использование функции `take-all` предполагает совмещение действий двух первых функций: вернется и изображение, закодированное в base64 с ключом **`image-key`**, и данные.

6.9.5 ЗАПРОС К ВЕБ-КАМЕРЕ (WEBCAM)

Пример запроса:

```
<hdw-request type="webcam" function="take-photo" params="save-file" hdw-
params="id">
```

Доступны следующие функции:

1. `take-photo` — получить изображение.
2. `stop-video` — остановить видеопоток.
3. `start-video` — запустить видеопоток.
4. `start-multimedia` — запустить видеопоток специальным образом: при помощи плеера VLC.
5. `open` — открыть порт камеры.
6. `close` — закрыть порт камеры.

Для функции `take-photo` доступен параметр **`save-file`**, содержащий в значении путь для сохранения изображения. Если параметр не указан, то изображение не будет сохранено.

Для всех функций доступны следующие дополнительные параметры:

1. **`id`** — идентификатор веб-камеры из конфигуратора. Если не указан, берется первая камера.

Для работы с функцией `start-multimedia` нужно, чтобы был установлен медиаплеер VLC и в конфигураторе прописан url для видеопотока. Функционал действует только на Java 13 и выше. В результате поток камеры будет транслироваться средствами VLC, который передает видео более высокого качества, чем стандартная функция `start-video`.

6.9.6 ЗАПРОС К КОНТРОЛЛЕРУ ЗАМКОВ (LOCKER)

Пример запроса:

```
<hdw-request type="locker" function="open-cell" params="id">
```

Доступна функция `open-cell` — открыть ячейку.

Обязательный параметр:

1. **`id`** — идентификатор ячейки.

В контексте возвращается инпут-элемент **cell-open-result**. Возможные значения:

- **true** — ячейка открылась;
- **false** — не открылась.

Для ТПО 7 **false** не возвращается — если ячейку не удалось открыть, действие завершится ошибкой.

6.9.7 ЗАПРОС К КОНТРОЛЛЕРУ КОНТАКТОВ (RELAY)

Пример запроса:

```
<hdw-request type="relay" function="close" params="id" >
```

Доступны следующие функции:

1. `close` — замкнуть контакт.
2. `state` — получить статус контакта.

Возможные параметры:

1. **`id`** — номер контакта.

6.9.8 ЗАПРОС К УСТРОЙСТВАМ ВЫДАЧИ (CASH-DISPENSER)

Проверяет, может ли терминал выдать запрашиваемую сумму. Запрос осуществляется через переменную *sum-out,* в значении которой указывается сумма для выдачи (в рублях).

Пример запроса:

```
<set key="sum-out" value="100.00" />
<hdw-request type="cash-dispenser" function="calc" params="sum-out">
```

Доступны следующие функции:

1. `calc` — высчитывает, может ли терминал выдать указанную сумму.

Возвращается **calc-result** со следующими возможными значениями:

1. InputElement **calc-result = true** — может выдать всю сумму.

2. InputElement **calc-result = false** — не может выдать запрашиваемую сумму.

3. InputElement **calc-result = false** и InputElement **calc-sum = <сумма в рублях, которую возможно выдать>** — может выдать только часть от запрашиваемой суммы.

6.9.9 ЗАПРОС ОТПРАВКИ ФАЙЛОВ (SEND-PHOTO)

При работе клиента с веб-камерой, установленной на терминале, возможно создание множества похожих фотографий в рамках платежа. Чтобы избежать отправки ненужных файлов на сервер, реализован запрос `send-photo`. Запрос осуществляется через переменные, в значении которых указывается путь до фотографии.

Пример запроса:

```
<set key="image1" value="<путь до файла>"/>
<set key="image2" value="<путь до файла2>"/>
<hdw-request type="send-photo" function="offline" params="image1,image2">
```

Функция `offline` добавляет указанные файлы в будущую операцию.

В ответе вернутся новые элементы с постфиксом у ключа **-exists**:

- **1** — файл есть;
- **0** — файла нет;
- если нет ни одного постфикса, вернется исключение.

Пример ответа:

```
...
15:09:02,103  INFO   image1:image1 =
/home/ivanov/Java/atm_paydepot_13/images/2021/10/20/point1_20211020122427
```

```
_.jpg:/home/ivanov/Java/atm_paydepot_13/images/2021/10/20/
point1_20211020122427_.jpg
15:09:02,103  INFO   image1-exists:image1-exists = 1:1
15:09:02,103  INFO   image2:image2 =
/home/ivanov/Java/atm_paydepot_13/images/2021/10/20/point1_20211020174320
_.jpg:/home/ivanov/Java/atm_paydepot_13/images/2021/10/20/
point1_20211020174320_.jpg
15:09:02,103  INFO   image2-exists:image2-exists = 0:0
...
```

Все пути в запросе должны быть указаны корректно, поскольку если возникнет проблема с одним из файлов, то вернется ошибка для всех.

6.9.10 ЗАПРОС ИНФОРМАЦИИ О ККТ (PRINTER)

Пример запроса:

```
<hdw-request type="printer" function="kkm-info">
```

Доступна функция `kkm-info` — для получения информации о ККМ.

В ответе вернутся два элемента с ключами ShiftNum и CashCode и их значениями, помещенные в контекст сценария.

Ключи обозначают:

1. ShiftNum — номер смены ККТ.
2. CashCode — заводской номер ККТ.

Запрос поддерживается на ТПО с версии 7.115.

6.9.11 ЗАПРОС РАСПОЗНАВАНИЯ ЛИЦА КЛИЕНТА (VERIFY-MATCH)

Механизм распознавания лиц поддерживается ТПО 7. Для осуществления запроса используется функция `verify-match`. В параметрах функции указываются пути до файлов, которые нужно сопоставить: например, путь до изображения лица клиента и путь до фотографии предъявленного документа. Обе фотографии делаются при помощи камеры терминала, примеры экранов идентификации приведены в разделе 4.6.26, описание запроса к камере — в разделе 6.9.5.

Пример запроса матчинга лиц:

```
<hdw-request type="face-recognition" function="verify-match"
params=”save-file-1, save-file-2”>
```

В ответе вернутся следующие атрибуты:

1. **`score`** — возвращает величину сходства изображений, основной критерий при определении успешности матчинга. Значение представляет собой действительное число от 0 до 1. Значение **1.0** будет означать полное совпадение.

2. **`distance`** — расстояние между изображениями.

3. **`fa_r`** — значение FAR (вероятность ложного сопоставления). Соответствует расстоянию **`distance`**, взятому в качестве порога на расширенном LFW-тесте.

4. **`fr_r`** — значение FRR (вероятность ложного отказа в доступе к изображению). Соответствует расстоянию **`distance`**, взятому в качестве порога на расширенном LFW-тесте.

Тип всех атрибутов — double. Подробнее см. здесь.

Другой вариант запроса предполагает не только сравнение изображений, но и сохранение файлов на диске ТПО с последующей отправкой на сервер. При этом дополнительно сохраняется еще одна фотография — обрезанное фото лица клиента.

При необходимости сохранить файлы укажите в запросе 5 параметров: кроме путей до изображений, которые нужно сравнить, добавьте параметры, которые будут определять место на диске для их хранения.

Пример запроса с сохранением фотографий:

```
<set key="save-path-image1"
value="/home/terminal/atm/images/image1.png" />
<set key="save-path-image1cut"
value="/home/terminal/atm/images/image1cut.png" />
<set key="save-path-image2"
value="/home/terminal/atm/images/image3.png" />

<hdw-request type="face-recognition" function="verify-match"
params="save-file-1,save-file-2,save-path-image1,save-path-
image1cut,save-path-image2">
```

Полное описание работы функционала матчинга приведено в документе «Терминальное ПО 7 версии. Программное обеспечение «Процессинговый центр Pay-logic». Руководство пользователя».

6.9.12 МЕТОД ОЦЕНКИ ПРИНАДЛЕЖНОСТИ ЛИЦА РЕАЛЬНОМУ ЧЕЛОВЕКУ

Метод доступен в ТПО версии 7. Используется для того, чтобы понять, на экране фотографирования лица был реальный человек или фотография. Для осуществления запроса используется функция `check-real`. В параметрах функции указывается путь до изображения, для которого нужно выполнить оценку.

Пример запроса оценки вероятности принадлежности лица реальному человеку:

```
<hdw-request type="face-recognition" function="check-real" params="save-
file">
```

В ответе вернется атрибут:

1. **`real`** — числовое значение в диапазоне от 0 до 1, выражающее вероятность принадлежности лица реальному человеку.

Пример запроса в сценарии:

```
<!-- Проверяем фото на реальность -->
<hdw-request type="face-recognition" function="check-real" params="save-
file">
  <actions>
    <action type="success">
      <!-- если результат проверки меньше 0.X то значит FAKE - показываем
ошибку -->
      <if condition="real lt 0.8">
        <then>
          <dialog type="Info" title="Error" message="Face is not recognized
or not real. Take a portrait shot of you again" timeout="30"
default="okay">
            <actions>
              <action type="okay" title="OK">
                <!-- Останавливаем камеру -->
                <hdw-request type="webcam" function="stop-video">
                  <actions>
                    <action type="success">
                      <goto target="FACE_MATCH_ID_FACE" />
                    </action>
                    <action type="error">
                      <dialog type="Info" title="Error" message="Operation is
impossible. Please try again later" timeout="30" default="okay">
                        <actions>
                          <action type="okay" title="OK">
                            <cancel />
                          </action>
                        </actions>
                      </dialog>
                    </action>
                  </actions>
                </hdw-request>
              </action>
            </actions>
          </dialog>
        </then>
      </if>
    </action>
    <action type="error">
      <dialog type="Info" title="Error" message="Face is not recognized or
not real. Take a portrait shot of you again" timeout="30" default="okay">
```

```
      <actions>
       <action type="okay" title="OK">
        <!-- Останавливаем камеру -->
        <hdw-request type="webcam" function="stop-video">
         <actions>
          <action type="success">
           <goto target="FACE_MATCH_ID_FACE" />
          </action>
          <action type="error">
           <dialog type="Info" title="Error" message="Operation is
impossible. Please try again later" timeout="30" default="okay">
            <actions>
             <action type="okay" title="OK">
              <cancel />
             </action>
            </actions>
           </dialog>
          </action>
         </actions>
        </hdw-request>
       </action>
      </actions>
     </dialog>
    </action>
  </actions>
</hdw-request>
```

6.10 ПЕЧАТЬ ЧЕКОВ БЕЗ ОПЛАТЫ (<PRINT>)

В усовершенствованном модуле обработчика существует возможность печати чеков без оплаты. Требуется в случаях, когда клиенту необходимо распечатать информацию, например, из онлайн-запроса. Для этого используется действие `<print>`. Действие `<print>` обеспечивает возможность печати чеков на любом этапе сценария оплаты.

Структура элемента:

```
<print template=<notEmptyString>/>
```

Атрибуты:

1. **`template`** — указывается имя файла, содержащего необходимый шаблон. Обязательный атрибут.

Пример:

```
<print template="rapidapatt"/>
```

Внимание!
Шаблоны чеков должны располагаться в каталоге ***<корень ТПО>/templates/default/custom/*** в 5 версии ТПО или ***<корень ТПО>/res/templates/default/custom/*** в 7 версии ТПО.

Объекты:

1. **custom.titles** — описывает заголовки атрибутов платежа. Доступен в чеке через метод `get`. Пример:

```
$custom.titles.get("description"): $!custom.values.get("description")
```

6.11 ПЕРЕНАПРАВЛЕНИЕ (<REDIRECT>)

Действие `<redirect>` служит для перехода в другой сервис.

Структура действия:

```
<redirect target=<notEmptyString>
          params=<keyListType>/>
```

Атрибуты описаны в таблице 6.11.1.

Таблица 6.11.1 — Атрибуты действия `<redirect>`

Атрибут	Описание	Обяз.
target	Идентификатор сервиса, на который требуется переход	Да
params	Передаваемые переменные, перечисляются через запятую без пробелов	Нет

Пример:

```
<redirect target="102" params="ls,fio"/>
```

При переходе на сервис 102 передаются переменные **ls** и **fio**.

6.12 ПОСЛЕДОВАТЕЛЬНОСТИ ЧИСЕЛ (<SEQUENCE-GENERATOR>)

Действие `<sequence-generator>` предназначено для генерации уникальных последовательностей.

```
<sequence-generator type=[simple]
                    name=<notEmptyString25>
                    start=<notEmptyString>
                    end=<notEmptyString>
                    loop=[true|false]
                    no-increment=[true|false]
                    result-key=<notEmptyString25>
                    result-title=<notEmptyString25>/>
```

Дочерние элементы: `<actions>` (раздел 4.3).

Атрибуты описаны в таблице 6.12.1.

Таблица 6.12.1 — Атрибуты действия `<sequence-generator>`

Атрибут	Описание	Обяз.
`type`	Определяет тип последовательности чисел. На текущий момент доступно одно значение: **`simple`** — простая последовательность чисел	Да
`name`	Наименование переменной, которая будет сохранена в в виде файла *<корень ТПО>/resources/sequences/{name}*	Да
`start`	Начальное значение последовательности. Возможно указывать числа типа Long и строки. Если значение, начинается с буквы или #, то в контексте ищется переменная с соответствующим именем и её значение подставляется в атрибут. Если значение начинается с $, то $ обрезается и переменная ищется в параметрах	Да

Атрибут	Описание	Обяз.
	точки	
end	Конечное значение последовательности. Возможно указывать числа типа Long и строки. Если значение, начинается с буквы или #, то в контексте ищется переменная с соответствующим именем и её значение подставляется в атрибут. Если значение начинается с $, то $ обрезается и переменная ищется в параметрах точки	Да
loop	Определяет осуществлять ли генерацию последовательности циклически. Когда последовательность закончится, генерация начнется заново. Возможные значения: • **true** — осуществлять циклически; • **false** — не осуществлять циклически. В этом случае при достижении конца последовательности будет выполнено действие с типом **type="error"**	Нет
no-increment	Определяет увеличивать или нет последовательность. Возможные значения: • **true** — не увеличивать; • **false** — увеличивать	Нет
result-key	Ключ результирующей последовательности	Да
result-title	Наименование результирующей последовательности	Да

В примере генерируется последовательность, которая сохраняется в файл *<корень ТПО>/resources/sequences/sequence* и переменную **seq**.

Пример:

```
<?xml version="1.0" encoding="utf-8"?>
<scenario begin="init">
  <!--Создание экрана формирования данных-->
  <screen id="init" title="*" type="Void">
```

```
   <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="Далее">
    <!--Генерация последовательности чисел-->
    <sequence-generator type="simple" name="sequence" start="1" end="5"
                        loop="true" no-increment="false" result-
key="seq"                              result-title="Seq">
     <actions>
      <!--Описание поведения сценария, если генерация последовательности
завершилась успешно-->
       <action type="success">
        <!--Объявление переменной result-->
        <set key="result" value="success"/>
       </action>
       <!--Описание поведения сценария, если генерация последовательности
завершилась ошибкой-->
       <action type="error" title="Error">
        <!--Объявление переменной result-->
        <set key="result" value="error"/>
       </action>
       </actions>
     </sequence-generator>
     <!--Переход на экран оплаты-->
     <goto target="pay"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-→
    <action type="Prev" title="Назад">
     <goto target="previous"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки"Выход"-->
    <action type="Exit" title="Выход">
     <goto target="exit"/>
    </action>
   </actions>
  </screen>
</scenario>
```

6.13 ОБЪЯВЛЕНИЕ ПЕРЕМЕННОЙ (<SET>)

Действие <set> служит для объявления новой переменной. Структура действия:

```
<set key=<notEmptyString25>
     key-title=<notEmptyString25>
     original-value=<notEmptyString25>
     value=<string>
     value-title=<notEmptyString40>
     flags=<flags_specification>
     key-title-id=<resourceIdType>
     value-title-id=<resourceIdType>/>
```

Дочерние элементы: отсутствуют.

Атрибуты описаны в таблице 6.13.1.

Таблица 6.13.1 — Атрибуты действия <set>

Атрибут	Описание	Обяз.
key	Указывается InputElement, содержащий ключ, получаемого атрибута	Да
key-title	Указывается InputElement, содержащий наименование, получаемого атрибута	Нет
value	Указывается InputElement, содержащий значение, получаемого атрибута	Да
value-title	Указывается InputElement, содержащий значение получаемого атрибута, отображаемое на экране	Нет
flags	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет
original-value	Позволяет задать значение для поля **originalValue** атрибута платежа. Это поле хранит значение, которое	Нет

Атрибут	Описание	Обяз.
	ввел пользователь без учета суффиксов и префиксов	
key-title-id	Значение ключа, по которому в ресурсах хранится текстовка, которая будет подставлена в **key-title** при отправке атрибута платежа на сервер. Текстовки расположены в каталоге ***<корень ТПО>/res/module/i18n/*** в файле *input.properties*. Поддерживается локализация, например, файл с текстовками на английском языке будет называться *input_en.properties*. Доступен в ТПО 7, 5 версией не обрабатывается	Нет
value-title-id	Значение ключа, по которому в ресурсах хранится текстовка, которая будет подставлена в **value-title** при отправке атрибута платежа на сервер. Текстовки расположены в каталоге ***<корень ТПО>/res/module/i18n/*** в файле *input.properties*. Поддерживается локализация, например, файл с текстовками на английском языке будет называться *input_en.properties*. Доступен в ТПО 7, 5 версией не обрабатывается	Нет

Пример:

```
<set key="id3"
     key-title="Услуга"
     value="9"
     value-title="Мойка двери"/>
```

Инициализируется переменная **id3**, значение которой равно 9, отображаемое значение — Мойка двери.

Использование <set> с атрибутом **flags** позволяет устанавливать атрибуту текущее значение флага. Например,

```
<set key="id3"
     key-title="Услуга"
```

```
     value="9"
     value-title="Мойка двери"
     flags="0x04"/>
```

В данном примере атрибут **id3** не будет отображаться в атрибутах платежа в кабинете агента.

Используя <set> в сценарии возможно получить и использовать текущую дату.

Пример:

```
<set key="date"
     value="$now{dd.MM.yyyy HH:mm:ss}"
     value-title="$now{dd.MM.yyyy HH:mm:ss}" />
```

В качестве **key** атрибута можно указать любое значение, например, **date**, **id4** и т. п. В **value** указывается дата в необходимом формате: DD — день, MM — месяц, yyyy — год. Возможный формат соответствует формату SimpleDateFormat (для 5 ТПО — https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html, для 7 ТПО — https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html). В **value-title** можно использовать любой нужный формат: **$now{dd.MM.yyyy HH:mm:ss}**, текущая дата и др.

6.14 СУММА ПЛАТЕЖА БЕЗ КОМИССИИ (<SET-SUM>)

Действие `<set-sum>` используется, если необходимо передать значение числовой переменной в сумму платежа без учета комиссии, например, если клиент вводит сумму платежа с квитанции.

Структура действия:

```
<set-sum from=<notEmptyString25>
         type=[units|decimals]/>
```

Дочерние элементы: отсутствуют.

Атрибуты описаны в таблице 6.14.1.

Таблица 6.14.1 — Атрибуты действия `<set-sum>`

Атрибут	Описание	Обяз.
`from`	Переменная, из которой передается значение	Да
`type`	Тип передаваемого значения. Возможные значения: • **`units`** — передается сумма в рублях; • **`decimals`** — передается в копейках	Да

Пример:

```
<set-sum from="sum"
         type="units"/>
```

В данном примере на экран вноса денежных средств передается значение переменной **`sum`** в рублях.

6.15 ОГРАНИЧЕНИЯ СУММ (<SET-SUM-LIMIT>)

Действие `<set-sum-limit>` используется для ограничения максимальной и минимальной зачисленных сумм. Обрабатывается ТПО 5, 7 версий, РМА 6. Лимиты, установленные в параметра, переопределяют лимиты из настроек сервиса. Лимиты считаются по сумме зачисленной.

Структура действия:

```
<set-sum-limit from=<notEmptyString25>
               type=[units|decimals]
               target=[max|min|max-deposit|min-limit]/>
```

Дочерние элементы: отсутствуют.

Атрибуты `from`, `type` аналогичны атрибутам элемента `<set-sum>` (раздел 6.14), **`target`** — устанавливает ограничение по суммам. Обязательный атрибут. Возможные значения:

1. **`max`** — ограничение максимальной зачисленной суммы.
2. **`min`** — ограничение минимальной зачисленной суммы.
3. **`max-deposit`** — ограничение максимальной вложенной суммы.
4. **`min-limit`** — ограничение минимального номинала первой вносимой купюры/монеты.

В примере ниже зачисленная сумма не может быть меньше значения переменной **`payamount`**.

Пример:

```
<set-sum-limit from="payamount"
```

```
                type="units"
                target="min"/>
```

В примере ниже первая вносимая купюра не может быть меньше 50, вложенная сумма не может быть больше 500. То есть первой можно вложить купюру номиналом или 50, или 100, или 500, но при этом, например, если внесено 460, то купюру номиналом 50 вложить уже невозможно, так как вложенная сумма будет равна 510, что противоречит ограничению **max-deposit**.

Пример:

```
<set key="max-deposit1" key-title="max-deposit1" value="500"
     value-title="500" />
<set key="min-limit1" key-title="min-limit1" value="50"
     value-title="50" />
<set-sum-limit from="max-deposit1" type="units" target="max" />
<set-sum-limit from="min-limit1" type="units" target="min" />
```

Если для сервиса или пункта меню установлена комиссия, то для корректной работы <set-sum-limit>, необходимо прописать в сценарии <verify type="sumpurchase"/>.

6.16 КОНКАТЕНАЦИЯ СТРОКОВЫХ ПЕРЕМЕННЫХ (<STR>)

Действие `<str>` служит для выполнения конкатенации строковых переменных. Доступны два варианта использования.

Структура действия:

1 вариант:

```
<str operation=<string>
     result-key=<keyType>
     result-title=<notEmptyString40>
     flags=<flags_specification>/>
```

2 вариант:

```
<str format=<string>
     result-key=<keyType>
     result-title=<notEmptyString40>
     flags="0x01"/>
```

Дочерние элементы: отсутствуют.

Атрибуты в 1 варианте аналогичны атрибутам элемента `<math>`, раздел 6.3.

В одном действии можно совершить только одну операцию конкатенации с двумя переменными. Действие работает только с переменными. Если к переменной необходимо прибавить произвольную строку, то следует воспользоваться тегом `<verify>`.

Пример:

```
<str operation="ls + sum"
     result-key="id1"
     result-title="Данные квитанции"
     flags="0x01"/>
```

Здесь переменная **id1** представляет собой строку, в которой началом является значение переменной **ls**, а концом значение переменной **sum**.

Во 2 варианте использования атрибуты **result-key**, **result-title**, **flags** аналогичны атрибутам 1 варианта. В атрибуте **format** задается строка, объединяющая произвольный текст и переменные. Для вывода значений переменных используется синтаксис **${ключ атрибута}**.

Пример:

```
<str format="${fio}, проживающему по адресу ${address} выдан кредит на
     сумму ${sum}"
     result-key="stage"
     result-title="stage"
     flags="0x01"/>
```

Пример (7 версия ТПО):

```
<scenario xmlns="http://pay-logic.ru" begin="1screen">
<!--Создание экрана ввода данных с несколькими полями-->
 <screen type="group" title ="Введите данные" id="1screen">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский без возможности
переключения языка, максимальное количество вводимых символов 60. Название
поля ввода: "ФИО плательщика"-->
   <text-field id="fio" title="ФИО плательщика" max-len="60"
               keyboard="any:[ru,symb]:upper:true">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules>
      <rule regex="^[а-яА-Яёё-]{2,20}[\s]{1}[а-яА-Яёё-]{2,20}[\s]{1,3}
                  [а-яА-ЯЁё-]{2,20}([\s]{1}[а-яА-ЯЁё-]{2,20})?$"/>
     </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите ФИО плательщика]]> </help>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и буквенная клавиатура, язык ввода – русский без возможности
```

```
переключения языка, максимальное количество вводимых символов 254. Название
поля ввода: "Адрес плательщика"-->
   <text-field id="address" title="Адрес плательщика" max-len="254"
               keyboard="any:[ru,symb]:upper:true">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules>
      <rule regex="^.{3,254}$"/>
     </rules>
    </validator>
    <!--Подсказка, отображается на экране-->
    <help> <![CDATA[<html>Введите адрес плательщика]]> </help>
   </text-field>
   <!--Создание поля ввода числа в десятичном формате через точку. Название
поля ввода: "Сумма кредита"-->
   <numeric-field id="sum" title="Сумма кредита" format="6.2">
    <!--Подсказка, отображается на экране-->
    <help>
     <![CDATA[<html>Введите сумму кредита<br>]]>
    </help>
    <verify>
     <range begin="10" end="999999"/>
    </verify>
   </numeric-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии "Далее"-->
   <action type="Next" title="Далее">
    <!--Конкатенация строковых переменных-->
    <str format="${fio}, проживающему по адресу ${address} выдан кредит на
                 сумму ${sum}" result-key="stage" result-title="stage" />
    <goto target="confirm"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии "Назад"-->
   <goto-action type="Prev" title="Назад" target="previous"/>
   <!--Описание поведения сценария оплаты при нажатии "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
 <!--Создание экрана подтверждения данных-->
 <screen type="confirm" decor="long" title ="Подтверждение данных платежа"
         id="confirm">
  <!--Список атрибутов, отображаемых на экране подтверждения-->
```

```
  <fields list="stage"/>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии "Далее"-->
   <goto-action type="Next" title="Далее" target="pay"/>
   <!--Описание поведения сценария оплаты при нажатии "Назад"-->
   <goto-action type="Prev" title="Назад" target="1screen"/>
   <!--Описание поведения сценария оплаты при нажатии "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
</scenario>
```

6.17 ФОРМИРОВАНИЕ СОСТАВНОГО АТРИБУТА (<TEXT-FORMAT-TABLE>)

Действие `<text-format-table>` позволяет создавать новый атрибут (фактически строку), состоящий из других атрибутов сценария, и определять правила форматирования при отображении на экране подтверждения. Действие `<text-format-table>` работает с nestedData (раздел 7). Если представить nestedData таблицей, то `<text-format-table>` обрабатывает строки таблицы (конкатенация значений InputElement).

Структура действия:

```
<text-format-table key=<notEmptyString25>
                   template=<notEmptyString>
                   input-data=<notEmptyString25>
                   filter=<notEmptyString25>
                   default-value=<notEmptyString25>/>
```

Дочерние элементы: отсутствуют.

Атрибуты описаны таблице 6.17.1.

Таблица 6.17.1 — Атрибуты действия `<text-format-table>`

Атрибут	Описание	Обяз.
key	Ключ создаваемого атрибута	Да
template	Определяет значения каких атрибутов будут входить в сформированный атрибут, например, *template="${name}:${summ} руб.* будет иметь вид *название_позиции:сумма.* В 5 версии ТПО поддерживается html-форматирование	Да
input-data	Атрибут, указывающий на переменную, которая	Да

Атрибут	Описание	Обяз.
	содержит данные для обработки. В выборку попадают только те значения, которые удовлетворяют условию из атрибута **filter**	
filter	Правило, согласно которому будут отбираться элементы из **input-data**. Возможные варианты: • **filter="selected="** — пустое значение InputElemet в столбце selected; • **filter="selected=1"** — равно 1; • **filter="selected"** — пустое	Нет
default-value	Если какой-то элемент атрибута **template** не определен для какой-то позиции, то подставится значение данного атрибута	Нет

Пример:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- text-format-table s -->
<scenario begin="init">
  <screen id="init" title="*" type="Void">
    <actions>
      <action type="Next" title="Далее">
        <!--Объявление переменных-->
        <set key="id" value="1"/>
        <set key="name" value="PHP"/>
        <set key="price" value="33.44"/>
        <!--Упаковка набора элементов-->
        <pack key="book1" type="data" elements="id,name,price"/>
        <!--Объявление переменных-->
        <set key="id" value="2"/>
        <set key="name" value="JavaScript"/>
        <set key="price" value="15"/>
        <!--Упаковка набора элементов-->
        <pack key="book2" type="data" elements="id,name,price"/>
        <!--Объявление переменных-→
        <set key="id" value="3"/>
        <set key="name" value="Perl"/>
        <set key="price" value="5"/>
        <!--Упаковка набора элементов-->
        <pack key="book3" type="data" elements="id,name,price"/>
```

```
    <!--Объявление переменных-->
    <set key="id" value="1"/>
    <set key="name" value="Василий"/>
    <set key="age" value="20"/>
    <!--Упаковка набора элементов-->
    <pack key="user1" type="data" elements="id,name,age"/>
    <!--Объявление переменных-->
    <set key="id" value="2"/>
    <set key="name" value="Мария"/>
    <set key="age" value=""/>
    <!--Упаковка набора элементов-->
    <pack key="user2" type="data" elements="id,name,age"/>
    <pack key="mixedNestedData" type="nested"
          elements="book1,book2,book3,user1,user2"/>
    <!--Удаление переменных из контекста-->
    <clear-except keys="mixedNestedData"/>

    <!--Получение итоговой суммы группового платежа-->
    <sum key="price" result-key="sum" input-data="mixedNestedData"
         filter="price!="/>
    <!--Форматирование составного атрибута-->
    <text-format-table key="allRows" template="id=${id}, name=${name}; "
                       input-data="mixedNestedData"/>
    <text-format-table key="book" template="id=${id}, name=${name}, price=${price}"
                       input-data="mixedNestedData" filter="id=3;name=Perl"/>
    <text-format-table key="user" template="id=${id}, name=${name}, age=${age}"
                       input-data="mixedNestedData" filter="id=2;age"
                       default-value="none"/>

    <goto target="confirm"/>
  </action>
  <!--Описание поведения сценария оплаты при нажатии "Назад"-->
  <action type="Prev" title="Назад">
    <goto target="previous"/>
  </action>
  <!--Описание поведения сценария оплаты при нажатии "Выход"-->
  <action type="Exit" title="Выход">
    <goto target="exit"/>
  </action>
 </actions>
</screen>

<!--Создание экрана подтверждения-->
<screen id="confirm" title="Проверка" type="Confirm">
 <!--Список атрибутов, отображаемых на экране подтверждения-->
 <fields list="allRows,book,user,sum"/>
 <actions>
  <!--Описание поведения сценария оплаты при нажатии "Далее"-->
```

```
    <action type="Next" title="Далее">
      <!--Объявление переменной-->
      <set key="id1" value="test"/>
      <goto target="pay"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии "Назад"-->
    <action type="Prev" title="Назад">
      <goto target="previous"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии "Выход"-->
    <action type="Exit" title="Выход">
      <goto target="exit"/>
    </action>
   </actions>
  </screen>
</scenario>
```

6.18 ПРЕДЗАПОЛНЕНИЕ ДАННЫМИ ИЗ ОКРУЖЕНИЯ (<LOAD>)

Действие `<load>` предназначено для создания переменной со значением, получаемым из атрибутов объектов ПС. В PMA поддержка реализована в версии 6.33.

Структура действия:

```
<load from=<notEmptyString25>
      key=<notEmptyString25>
      title=<notEmptyString40>
      flags=<flags_specification>
      key-title-id=<notEmptyString40>/>
```

Дочерние элементы: отсутствуют.

Атрибут `flags` описан в разделе 5.5, **`key-title-id`** аналогичен атрибуту `<set>` (раздел 6.13), остальные атрибуты описаны в таблице 6.18.1.

Таблица 6.18.1 — Атрибуты действия `<load>`

Атрибут	Описание	Обяз.
`from`	Указывает поле объекта из числа доступных в контексте сценария, значение которого будет присвоено создаваемой переменной. Доступные объекты и их поля описаны в разделе 9.	Да
`key`	Ключ создаваемой переменной. Если не задан, то будет использован ключ поля, указанного в атрибуте **`from`**	Нет
`title`	Заголовок создаваемой переменной. Если не задан, то будет использован заголовок поля, указанного в атрибуте **`from`**	Нет

Атрибут	Описание	Обяз.
flags	Флаги, дополнительные свойства для получаемого атрибута платежа из данного поля, могут задаваться как числом (битовой маской), так и символьными константами. Описаны в разделе 5.5	Нет
key-title-id	Значение ключа, по которому в ресурсах хранится текстовка, которая будет подставлена в **key-title** при отправке атрибута платежа на сервер. Текстовки расположены в каталоге ***<корень ТПО>/res/module/i18n/*** в файле *input.properties*. Поддерживается локализация, например, файл с текстовками на английском языке будет называться *input_en.properties*. Доступен в ТПО 7, 5 версией не обрабатывается	Нет

Пример:

```
<load from="point.id" key="id1" title="Номер точки"/>
```

В данном примере создается переменная **id1** со значением, взятым из атрибута **point.id** точки процессинга и названием «Номер точки».

6.19 ОТКРЫТИЕ HTML-СТРАНИЦЫ (<OPEN-URL>)

Действие `<open-url>` предназначен для открытия html-страниц в электронном кошельке.

Структура действия:

```
<open-url url=<urlType>
          target=[BLANK|SELF|PARENT]
          method=[POST|GET]
          encoding=<notEmptyString15>>
       <params id="mobile" from="id"/>
       ...
</open-url>
```

Дочерние элементы: `<params>`.

Атрибуты описаны в таблице 6.19.1.

Таблица 6.19.1 — Атрибуты действия `<open-url>`

Атрибут	Описание	Обяз.
`url`	Открываемый URL	Да
`target`	Определяет, где открыть документ(обязателен). Возможные значения: • **`BLANK`** — загрузить в новом окне браузера; • **`SELF`** — загрузить в текущем окне; • **`PARENT`** — загрузить в родительском фрейме, если фреймов нет, то работает как **`SELF`**	Да
`method`	Тип запроса. Возможные значения: **`POST`** (по умолчанию), **`GET`**	Нет
`encoding`	Кодировка, значение по умолчанию — UTF-8	Нет

Элемент `<params>` определяет параметры запроса.

Атрибуты:

1. **`id`** — ключ параметра запроса.

2. **`from`** — ключ параметра контекста сценария, значение которого будет добавлено в запрос (под ключом указанным в атрибуте **`id`**).

Пример:

```
<open-url url="http://soft-logic.ru/test"
          target="SELF"
          method="POST"
          encoding="UTF-8">
      <param id="mobile" from="id"/>
      <param id="name" from="name"/>
</open-url>
```

6.20 ВКЛЮЧЕНИЕ ДАННЫХ ИЗ ВНЕШНЕГО ФАЙЛА (<XI:INCLUDE>)

Включение данных из внешнего файла в 5 (7im), 7 версии ТПО осуществляется с использованием элемента <xi:include>. В корневом теге сценария необходимо подключить спецификацию Xinclude, указав конструкцию xmlns:xi="http://www.w3.org/2001/Xinclude". Поиск файла-источника ведется в корневом каталоге ТПО или возможно указать полный путь до файла-источника. Файл-источник поддерживает те же правила локализации, что и сценарии.

Пример:

```
<scenario xmlns="http://pay-logic.ru"
        xmlns:xi="http://www.w3.org/2001/XInclude" begin="01screen">
 <!--Создание экрана ввода числовой и текстовой информации-->
 <screen type="numeric" title="Пин-код карты сдачи"
         message="Введите пин-код карты сдачb" id="01screen">
 <fields>
   <!--Включение в сценарий данных из внешнего файла--
   <xi:include href="include.xml"/>
 </fields>
 <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
     <goto target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <action type="Prev" title="Назад">
     <goto target="previous"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <action type="Exit" title="Выход">
     <goto target="exit"/>
   </action>
   </actions>
 </screen>
</scenario>
```

Содержимое файла *include.xml*:

```
<!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 16. Название поля
ввода: "Пин-код карты сдачи"-->
<text-field id="id1" title="Пин-код карты сдачи" max-len="16"
            keyboard="Digital">
    <!--Регулярное выражение для валидации вводимого пин-кода-->
    <validator type="regex">
      <rules> <rule regex="^\d{16}$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого номера на
экране-->
    <formatter type="regex">
      <rules default="****-****-****-****"/>
    </formatter>
    <!--Подсказка, отображается на экране-->
    <help> Введите пин-код карты сдачи </help>
</text-field>
```

В примере файл-источник расположен в корневом каталоге ТПО.

Возможно указать полный путь до файла-источника и расположить его в другом каталоге, например:

```
<xi:include
href="/home/user/Work/atm7/res/module/input/advanced/include.xml"/>
```

6.21 ПОЛУЧЕНИЕ ДАННЫХ С КАРТОЧНОГО МОДУЛЯ (<CARD-MODULE-TASK>)

Для ТПО 5 в сценарии доступна конструкция `<card-module-task>`, которая позволяет возвращать следующие элементы:

1. **masked-pan** — маскированный pan карты, например, 1234********6789.
2. **hash-pan** — sha-1 хеш номера карты.
3. **balance** — баланс в виде «123.45 RUB».
4. **card-expiry** — срок действия карты в формате YYMM.
5. **inn** — результат проверки «Своя/чужая карта». Возможные значения:
    1) 0 — карта принадлежит чужому банку;
    2) 1 — карта принадлежит своему банку.

Элементы **masked-pan** и **hash-pan** будут присутствовать всегда, а остальные input-элементы будут зависеть от типа операции.

Конструкция `<card-module-task>` имеет атрибут **operation**. Сейчас обрабатывается **operation=«balance».** Обрабатывается также **operation=«inn»**.

Пример использования:

```
<scenario xmlns="http://pay-logic.ru" begin="first_screen">
 <!--Создание экрана формирования данных-->
 <screen type="Void" decor="simple" id="first_screen" title="">
  <fields/>
  <actions>
  <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="ДАЛЕЕ">
    <!--Получение данных с карточного модуля-->
    <card-module-task operation="balance">
     <actions>
      <!--Описание поведения сценария оплаты в случае успешного получения
данных с карточного модуля-->
      <action type="success">
       <goto target="r_screen"/>
      </action>
     </actions>
    </card-module-task>
   </action>
  </actions>
 </screen>
 <!--Создание экрана подтверждения-->
 <screen type="Confirm" decor="simple" title="Result screen"
         id="r_screen">
  <!--Список атрибутов, отображаемых на экране подтверждения-->
  <fields list="masked-pan,hash-pan,balance"/>
   <actions>
    <!--Описание поведения сценария оплаты при нажатии "Далее"-->
    <action type="Next" title="Далее">
     <goto target="pay"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии "Назад"-->
    <action type="Prev" title="Назад">
     <goto target="exit"/>
    </action>
   </actions>
  </screen>
</scenario>
```

У `<card-module-task>` обрабатывается только `<action>` с **type=«success»**, action'ы с типами **«error»**, **«exception»** в данном случае обрабатывать не нужно, так как этим занимается карточный модуль.

Пример проверки «Своя/чужая карта»:

```
<card-module-task operation="iin">
...
</card-module-task>
```

Доступна карточная операция **operation=«mti»**. При ее использовании обязательно указание дополнительного параметра **mti-code**, который является номером MTI-запроса и определяется хостом банка.

Пример использования:

```
<card-module-task operation="mti" mti-code="10">
```

В примере выше показан MTI-запрос с номером «**mti-code=10**», который позволяет получить данные о номере счёта клиента.

Данные ответа MTI-запроса складываются в контекст модуля ввода данных в виде InputElement'ов. Ключи InputElement'ов, которые попадут в контекст обработчика, соответствуют тем, что были получены от хоста в поле «**g**».

Конструкции `<card-module-task>` имеет атрибут **eject**, который позволяет вернуть карту сразу после выполнения конструкции. По умолчанию атрибут имеет значение **false**. Для того, чтобы настроить возврат карты, установите для атрибута **eject** значение **true**, как в следующем примере:

```
<card-module-task operation="balance" eject="true">
```

6.22 РАСЧЕТ ХЕША (<HASH>)

Для ТПО 5, 7 в сценарии доступна конструкция `<hash>`, которая позволяет возвращать хеш от значения.

Структура действия:

```
<hash src-key=<notEmptyString40>
      method=MD5|SHA256|SHA-256|SHA1|SHA-1]
      target-key=<notEmptyString40>/>
```

Атрибуты описаны в таблице 6.22.1.

Таблица 6.22.1 — Атрибуты действия `<hash>`

Атрибут	Описание	Обяз.
src-key	Переменная, из которой берется значение. Возможно указать произвольную строку	Да
method	Определяет, алгоритм расчета хеша. Возможные значения: • **MD5** — 128-битный алгоритм хеширования (Message Digest 5). Используется по умолчанию; • **SHA256**, **SHA-256** — однонаправленная хеш-функция SHA-256; • **SHA1**, **SHA-1** — алгоритм криптографического хеширования SHA-1 (Secure Hash Algorithm 1)	Нет
target-key	Переменная, в которую записывается значение. Если не указано, то значение записывается в переменную, указанную в **src-key**	Нет

Пример:

```
<hash src-key="h1" method="MD5" target-key="h3"/>

<hash src-key="hash string" target-key="h2"/>

<hash src-key="h1"/>

<hash src-key="h4" method="SHA-256" target-key="h5"/>

<hash src-key="h4" method="SHA256"/>

<hash src-key="h6" method="SHA-1" target-key="h7"/>

<hash src-key="h6" method="SHA1"/>
```

6.23 ВЫЗОВ СКРИПТА (<EXECUTE-SCRIPT>)

Для ТПО 5, 7 в сценарии доступна конструкция `<execute-script>`, которая позволяет вызвать скрипт, размещенный в каталоге ***<корень ТПО>/bin/***.

Структура действия:

```
<execute-script script=<notEmptyString25>
                params=<keyListType>
      <actions>
          ...
      </actions>
</execute-script>
```

Атрибуты описаны в таблице 6.23.1.

Таблица 6.23.1 — Атрибуты действия `<execute-script>`

Атрибут	Описание	Обяз.
script	Наименование вызываемого скрипта. Указывается без расширения. Для ОС Windows можно использовать в качестве скрипта файлы с расширениями: .cmd, .bat, для ОС семейства Linux — .sh или без указания расширения. Скрипт должен находиться в каталоге ***<корень ТПО>/bin/***	Да
params	Список, передаваемых в скрипт параметров. Значения из **params** передаются в скрипт в том порядке, в котором они указаны. Если **params="$all"** передаются все элементы. Для запуска скрипта без параметров атрибут **params** должен отсутствовать В скрипте передаваемые через **params** атрибуты	Нет

Атрибут	Описание	Обяз.
	вызываются с помощью конструкции вида **${НОМЕР_АТРИБУТА}** (нумерация начинается с 1)	

Пример:

```
<execute-script script="scriptName" params="id1,id2">
      <actions>
          <action type="success">
                // получили код 0
          </action>
          <action type="error">
                // получили не код 0, значение кода ошибки в #error-code
          </action>
          <action type="exception">
                // нет скрипта, не смогли его выполнить
          </action>
      </actions>
</execute-script>
```


Web: https://soft-logic.ru/
Mail: info@soft-logic.ru

6.24 ЧТЕНИЕ ТРЕКОВ КАРТЫ (<READ-CARD-TRACK>)

Для ТПО 5 и 7 в сценарии доступна конструкция `<read-card-track>`, которая позволяет считывать треки карты.

Структура действия:

```
<read-card-track key=<keyType>
                 key-title=<notEmptyString40>
                 track=[1|2|3]>
     <actions>
          <action type="success">
               ...
          </action>
          <action type="error">
               ...
          </action>
     </actions>
</read-card-track>
```

Атрибуты описаны в таблице 6.24.1.

Таблица 6.24.1 — Атрибуты действия `<read-card-track>`

Атрибут	Описание	Обяз.
key	Ключ атрибута, в который будет записано значение трека	Да
key-title	Наименование атрибута, в который будет записано значение трека	Нет
track	Номер считываемого трека	Да

Пример использования:

```
<screen type="Void" decor="simple" id="first_screen" title="">
 <fields/>
 <actions>
  <action type="Next" title="ДАЛЕЕ">
   <read-card-track key="#track" key-title="Номер карты"
          track="2">
    <actions>
     <action type="success">
      ...
     </action>
     <action type="error">
      ...
     </action>
    </actions>
   </read-card-track>
  </action>
 </actions>
</screen>
```


Web: https://soft-logic.ru/
Mail: info@soft-logic.ru

6.25 ЧТЕНИЕ ДАННЫХ ИЗ XML-ФАЙЛОВ (<XPATH>)

Для ТПО 5 и 7 в сценарии доступна конструкция `<xpath>`, которая позволяет извлекать данные из xml-файлов.

Структура действия:

```
<xpath path=<<string>>>
      <select expression=<string>>
          <item key=<keyType>
                key-title=<notEmptyString40>
                value=<string>
                value-title=<notEmptyString40>
                flags=<flags_specification>
                key-title-id=<resourceIdType>
                value-title-id=<resourceIdType>/>
          ...
          <item .../>
      </select>
      <select-nodes expression=<string>>
          <item key=<keyType>
                key-title=<notEmptyString40>
                value=<string>
                value-title=<notEmptyString40>
                flags=<flags_specification>
                key-title-id=<resourceIdType>
                value-title-id=<resourceIdType>/>
      </select-nodes>
</xpath>
```

Дочерние элементы: `<select>`, `<select-nodes>`. В одном `<xpath>` возможно использовать несколько `<select>` и `<select-nodes>` одновременно.

Атрибуты `<xpath>` приведены в таблице 6.25.1.

Таблица 6.25.1 — Атрибуты действия `<xpath>`

Атрибут	Описание	Обяз.
path	Путь до xml-файла относительного корневого каталога ТПО. Например, если файл *res.xml* расположен в корневом каталоге ТПО, то **path="res.xml"**. Если файл *res.xml* расположен в каталоге ***<корень ТПО>/resources***, то **path="resources/res.xml"**	Да

Элемент <select> позволяет извлечь только один XML-элемент из файла.

Элемент <select-nodes> позволяет извлечь весь массив данных из XML-файла.

Дочерние элементы <select>, <select-nodes>: <item>. В <select> возможно использовать несколько <item>, в <select-nodes> только один элемент <item>.

Атрибуты <select>, <select-nodes> приведены в таблице 6.25.2.

Таблица 6.25.2 — Атрибуты <select>, <select-nodes>

Атрибут	Описание	Обяз.
expression	Путь до элемента от корня XML-файла. Например, **expression=" /response/users/user"**. В **expression** возможно использовать фильтры для выбора данных, фильтр заключается в квадратные скобки. Например, **expression=" /response/users/user[@account='1']"**. Для обращения к атрибуту используется синтаксис **"@ключ атрибута"**, то есть, например, **@account**. Синтаксис выражений, которые возможно использовать в атрибуте **expression** подробно описан по адресу: https://www.w3schools.com/xml/xpath_syntax.asp	Да

Атрибуты элемента <item> приведены в таблице 6.25.3.

Таблица 6.25.3 — Атрибуты элемента <item>

Атрибут	Описание	Обяз.
key	Ключ атрибута, в который будет сохранено извлеченное значение	Да
key-title	Отображаемое название атрибута	Нет
value	Реальное значение атрибута	Да
value-title	Отображаемое значение атрибута	Нет
flags	Позволяет установить флаг атрибуту. Подробнее в разделе 5.5	Нет
key-title-id	Значение ключа, по которому в ресурсах хранится текстовка, которая будет подставлена в **key-title** при отправке атрибута платежа на сервер. Подробнее в разделе 5.3	Нет
value-title-id	Значение ключа, по которому в ресурсах хранится текстовка, которая будет подставлена в **value-title** при отправке атрибута платежа на сервер. Подробнее в разделе 5.3	Нет

Для извлечения значения параметра внутри <item> используется синтаксис **"@{ключ атрибута}"**, например, **key="@{name}"**.

Пример 1:

В XML-файле в одном параметре элемента содержится название атрибута, а в другом — значение. Как правило, файлы такой структуры являются результатом выполнения внешних скриптов.

Пример XML-файла:

```
<?xml version="1.0" encoding="UTF-8"?>
<response code="0" message="Success">
 <params>
   <param name="datetime" value="20170215115706"/>
   <param name="state" value="0"/>
   <param name="transaction" value="be87499e80644a449207a3626fe3433e"/>
   <param name="terminal" value="00000000000000000352701060587976"/>
   <param name="start-balance" value="00695300"/>
   <param name="balance" value="00697100"/>
   <param name="can" value="6378990000022652"/>
   <param name="exp-date" value="20210330"/>
   <param name="load-trans" value="30"/>
 </params>
</response>
```

Для извлечения таких данных целесообразно использовать <select-nodes>. В примере <select-nodes> извлекает данные из элементов <param> XML-файла. Атрибуты сохраняются с ключом, извлеченным из параметра **name** XML-элемента, и значением, извлеченным из параметра **value** XML-элемента, то есть в примере **datetime= 20170215115706**, отображаемый заголовок атрибута «key-title.datetime»; **state=0**, отображаемый заголовок атрибута «key-title.state»; **transaction=be87499e80644a449207a3626fe3433e**, отображаемый заголовок атрибута «key-title.transaction» и т. д. Элемент <select> в примере извлекает из элемента <response> XML-файла **code** и **message**. То есть к платежу добавляются атрибуты: **key=0**, отображаемый заголовок атрибута «Результат выполнения» и **message=Success**, отображаемый заголовок атрибута «Сообщение».

Пример сценария:

```
<scenario begin="input_ls">
 <!--Создание экрана ввода данных с несколькими полями-->
 <screen type="group/numeric" decor="simple"
         title ="Введите лицевой счет" id="input_ls">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 10. Название поля
ввода: "Лицевой счет"-->
   <text-field id="id1" title="Лицевой счет" max-len="10"
```

```
                keyboard="Digital" >
      <!--Регулярное выражение для валидации вводимого лицевого счета-→
      <validator type="regex">
        <rules>
          <rule regex="^\d{1,10}$"/>
        </rules>
        </validator>
      <!--Регулярное выражение для форматирования вводимого номера на
экране-->
        <formatter type="regex">
          <rules default="*** *** ****"/>
        </formatter>
      </text-field>
    </fields>
    <actions>
      <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
      <action type="Next" title="ДАЛЕЕ">
        <!--Чтение данных из файла res.xml-->
        <xpath path="res.xml">
          <select expression="/response">
            <item  key="result" key-title="Результат выполнения"
                   value="@{code}" value-title="@{code}" flags="1"
                   key-title-id="agent.name.title"
                   value-title-id="agent.name.title.value" />
            <item key="message" key-title="Сообщение" value="@{message}"
                  value-title="@{message}" flags="1"
                  key-title-id="agent.name.title"
                  value-title-id="agent.name.title.value" />
          </select>
          <select-nodes expression="/response/params/param">
            <item  key="@{name}" key-title="key-title.@{name}"
                   value="@{value}" value-title="@{value}" flags="1"
                   key-title-id="agent.name.title"
                   value-title-id="agent.name.title.value" />
          </select-nodes>
        </xpath>
        <!--Печать чека-->
        <print template="paynet"/>
        <goto target="pay"/>
      </action>
      <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
      <action type="Prev" title="НАЗАД">
        <goto target="previous"/>
```

```
      </action>
      <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
      <action type="Exit" title="ВЫХОД">
        <goto target="exit"/>
      </action>
    </actions>
  </screen>
</scenario>
```

Пример 2:

XML-файл представляет собой справочник. Один XML-элемент содержит неограниченный перечень данных о каком-либо объекте.

Пример XML-файла:

```
<?xml version="1.0" encoding="UTF-8"?>
 <users>
   <user account="1" fio="ФИО1" address="Адрес1"/>
   <user account="2" fio="ФИО2" address="Адрес2"/>
   <user account="3" fio="ФИО3" address="Адрес3"/>
 </users>
</response>
```

В сценарии ниже первый элемент `<select>` извлекает XML-элемент `<user>`, параметр **`account`** которого равен **`1`** (**`[@account='1']`**). Второй элемент `<select>` извлекает XML-элемент `<user>`, параметр **`account`** которого равен введенному в сценарии номеру лицевого счета (**`id1`**). Для передачи значения атрибута в фильтре **`expression`** используется синтакис **`${ключ атрибута}`** — **`[@account='${id1}']`**. Каждый параметр XML-элемента `<user>` извлекается с помощью `<item>`. В примере:

1. Для XML-элемента `<user>`, атрибут **`account`** которого равен **`1`:**

   1) значение параметра **`account`** записывается в атрибут **`acc`** с отображаемым заголовком «ID Агента»;

   2) значение параметра **`fio`** записывается в атрибут **`fio`** с отображаемым заголовком «Фамилия»;

3) значение параметра **address** записывается в атрибут **address** с отображаемым заголовком «Адрес».

2. Для XML-элемент <user>, атрибут **account** которого равен введенному номеру лицевого счета (атрибут **id1**):

1) значение параметра **account** записывается в атрибут **acc2** с отображаемым заголовком «ID Агента2»;

2) значение параметра **fio** записывается в атрибут **fio2** с отображаемым заголовком «Фамилия2»;

3) значение параметра **address** записывается в атрибут **address2** с отображаемым заголовком «Адрес2».

Пример сценария:

```
<scenario begin="input_ls">
 <!--Создание экрана ввода данных с несколькими полями-->
 <screen type="group/numeric" decor="simple"
         title ="Введите лицевой счет" id="input_ls">
   <fields>
    <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 10. Название поля
ввода: "Лицевой счет"-->
    <text-field id="id1" title="Лицевой счет" max-len="10"
                keyboard="Digital" >
     <!--Регулярное выражение для валидации вводимого лицевого счета-->
     <validator type="regex">
      <rules>
       <rule regex="^\d{1,10}$"/>
      </rules>
     </validator>
   <!--Регулярное выражение для форматирования вводимого лицевого счета
на экране-->
     <formatter type="regex">
      <rules default="*** *** ****"/>
     </formatter>
    </text-field>
   </fields>
   <actions>
```

```
<!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
<action type="Next" title="ДАЛЕЕ">
 <!--Чтение данных из файла res.xml-->
 <xpath path="res.xml">
  <select expression="/users/user[@account='1']">
   <item  key="acc" key-title="ID Агента" value="@{account}"
          value-title="@{account}" flags="1"
          key-title-id="agent.name.title"
          value-title-id="agent.name.title.value" />
   <item  key="fio" key-title="Фамилия" value="@{fio}"
          value-title="@{fio}" flags="1"
          key-title-id="agent.name.title"
          value-title-id="agent.name.title.value" />
   <item  key="address" key-title="Адрес" value="@{address}"
          value-title="@{address}" flags="1"
          key-title-id="agent.name.title"
          value-title-id="agent.name.title.value" />
  </select>
  <select expression="/users/user[@account='${id1}']">
   <item  key="acc2" key-title="ID Агента2" value="@{account}"
          value-title="@{account}" flags="1"
          key-title-id="agent.name.title"
          value-title-id="agent.name.title.value" />
   <item  key="fio2" key-title="Фамилия2" value="@{fio}"
          value-title="@{fio}" flags="1"
          key-title-id="agent.name.title"
          value-title-id="agent.name.title.value" />
   <item  key="address2" key-title="Адрес2" value="@{address}"
          value-title="@{address}" flags="1"
          key-title-id="agent.name.title"
          value-title-id="agent.name.title.value" />
  </select>
 </xpath>
 <!--Печать чека-->
 <print template="paynet"/>
 <goto target="pay"/>
</action>
<!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
<action type="Prev" title="НАЗАД">
 <goto target="previous"/>
</action>
<!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
<action type="Exit" title="ВЫХОД">
```

```
      <goto target="exit"/>
    </action>
   </actions>
  </screen>
</scenario>
```

В XML-файле могут содержаться любые блоки данных, для извлечения которых необходимо указать соответствующий путь в атрибуте **`expression`** элементов `<select>` или `<select-nodes>`.

7 ГРУППОВЫЕ ОПЕРАЦИИ. КОРЗИНА ПЛАТЕЖЕЙ

7.1 ОБЩИЕ СВЕДЕНИЯ

Групповые операции позволяют клиенту в рамках одного сервиса реализовывать оплату нескольких услуг/товарных позиций и выдавать на каждую оплату отдельный чек. В терминологии процессинга для обозначения групповой операции используется понятие «Корзина платежей». Примером использования корзины платежей является покупка нескольких товаров в столовой: первое, второе, компот или выбор коммунальных услуг для оплаты из списка, сформированного по введенному номеру лицевого счета. Выбор всех позиций осуществляется на одном экране.

В процессинге существуют следующие типы атрибутов платежа:

1. **inputElement** — примитивный тип, атрибут платежа.
2. **Data** — составной элемент, содержит в себе набор inputElement-ов.
3. **NestedData** — представляет собой многострочный элемент, каждая строка которого представляет собой Data-элемент.

При формировании атрибутов платежа групповой операции используются объект типа NestedData.

Для сценариев с использованием корзины платежей определены теги, позволяющие проводить единовременно группу платежей, такие как `<payment-params>`, `<communal-prepare>`, `<pack>`, `<unpack>`, `<sum>`.

Элементы корзины имеют следующие предопределенные атрибуты:

1. **selected** — определяет, выбран ли элемент. Может принимать значения 1 (элемент выбран) и 0 (элемент не выбран).

2. **editable** — определяет, можно ли редактировать элемент. Может принимать значения 1 (элемент можно редактировать) и 0 (элемент нельзя редактировать).

3. **summ** — цена элемента.

4. **allow-change-state** — определяет доступность элемента для выбора. Может принимать значения 1 (элемент можно выбрать) и 0 (элемент нельзя выбрать).

5. **name** — задает наименование элемент.

6. **title** — заголовок элемента, если не указан, используется **name**. Если указаны оба, используется первый встречающийся. Выводится в качестве надписи на кнопке с элементом. Используется только в 7 версии ТПО, в 5 недоступен.

7. **comm** — сумма комиссии.

8. **#selected** — указание на выбранный элемент data внутри NestedData.

9. **#removed** — указание на удаляемый элемент data внутри NedtedData.

10. **#for-element** — цикл по каждому элементу из elements.

Рассмотрим пример использования корзины в сценарии. Первоначально на экране «VOID» формируются данные. С помощью элементов `<set>` задаются значения атрибутов, а затем упаковываются в элемент типа **data** с использованием `<pack>`.

После того, как созданы все элементы **data**, они упаковываются в корзину:

```
<pack type="nested" key="#services" elements="dt1,dt2,dt3,dt4,dt5,dt6,dt7"/>
```

Затем данные передаются на экран оплаты нескольких услуг «COMMUNAL» с помощью конструкции:

```
<fields key="#services"/>
```

Пример сценария:

```
<scenario xmlns="http://pay-logic.ru" begin="init_screen"
          begin-process="group_screen3">
  <!--Создание экрана формирования данных-->
  <screen type="Void" decor="simple" title ="Наполнение корзины"
           id="init_screen">
   <actions>
     <!--Описание поведения сценария оплаты при нажатии "Далее"-->
     <action type="Next" title="ДАЛЕЕ">
      <!--Объявление переменных-->
      <set key="selected" value="0"/>
      <set key="editable" value="0"/>
      <set key="summ" key-title="Цена" value="200" value-title="200" />
      <set key="allow-change-state" value="1"/>
      <set key="id3" key-title="Услуга" value="1"
           value-title="Заточка коньков" />
      <set key="name" key-title="Прайс" value="Заточка коньков - 200р."
           value-title="Заточка коньков - 200р." />
      <!--Упаковка набора элементов-->
      <pack type="data" key="dt1"
            elements="selected,editable,summ,allow-change-state,id3,name"/>
      <!--Объявление переменных-->
      <set key="selected" value="0"/>
      <set key="editable" value="0"/>
      <set key="summ" key-title="Цена" value="250" value-title="250" />
      <set key="allow-change-state" value="1"/>
      <set key="id3" key-title="Услуга" value="2"
           value-title="Заточка НОВЫХ коньков" />
      <set key="name" key-title="Прайс" value="Заточка НОВЫХ коньков - 250р."
           value-title="Заточка НОВЫХ коньков - 250р." />
      <!--Упаковка набора элементов-->
      <pack type="data" key="dt2"
             elements="selected,editable,summ,allow-change-state,id3,name"/>
      <!--Объявление переменных-->
      <set key="selected" value="0"/>
      <set key="editable" value="0"/>
      <set key="summ" key-title="Цена" value="200" value-title="200" />
      <set key="allow-change-state" value="1"/>
      <set key="id3" key-title="Услуга" value="3"
          value-title="Заточка хоккейных коньков" />
      <set key="name" key-title="Прайс"
          value="Заточка хоккейных коньков - 200р."
```

```
     value-title="Заточка хоккейных коньков - 200р." />
<!--Упаковка набора элементов-->
<pack type="data" key="dt3"
      elements="selected,editable,summ,allow-change-state,id3,name"/>
<!--Объявление переменных-→
<set key="selected" value="0"/>
<set key="editable" value="0"/>
<set key="summ" key-title="Цена" value="800" value-title="800" />
<set key="allow-change-state" value="1"/>
<set key="id3" key-title="Услуга" value="4"
     value-title="Заточка НОВЫХ хоккейных коньков" />
<set key="name" key-title="Прайс"
     value="Заточка НОВЫХ хоккейных коньков - 800р."
     value-title="Заточка НОВЫХ хоккейных коньков - 800р." />
<!--Упаковка набора элементов-->
<pack type="data" key="dt4"
      elements="selected,editable,summ,allow-change-state,id3,name"/>
<!--Объявление переменных-->
<set key="selected" value="0"/>
<set key="editable" value="0"/>
<set key="summ" key-title="Цена" value="100" value-title="100" />
<set key="allow-change-state" value="1"/>
<set key="id3" key-title="Услуга" value="5"
     value-title="Подготовка лезвия" />
<set key="name" key-title="Прайс" value="Подготовка лезвия - 100р."
     value-title="Подготовка лезвия - 100р." />
<!--Упаковка набора элементов-->
<pack type="data" key="dt5"
      elements="selected,editable,summ,allow-change-state,id3,name"/>
<!--Объявление переменных-->
<set key="selected" value="0"/>
<set key="editable" value="0"/>
<set key="summ" key-title="Цена" value="400" value-title="400" />
<set key="allow-change-state" value="1"/>
<set key="id3" key-title="Услуга" value="6"
     value-title="Правка лезвия" />
<set key="name" key-title="Прайс" value="Правка лезвия - 400р."
     value-title="Правка лезвия - 400р." />
<!--Упаковка набора элементов-->
<pack type="data" key="dt6"
      elements="selected,editable,summ,allow-change-state,id3,name"/>
<!--Объявление переменных-->
<set key="selected" value="0"/>
```

```
      <set key="editable" value="0"/>
      <set key="summ" key-title="Цена" value="150" value-title="150" />
      <set key="allow-change-state" value="1"/>
      <set key="id3" key-title="Услуга" value="7"
           value-title="Шлифовка лезвия" />
      <set key="name" key-title="Прайс" value="Шлифовка лезвия - 150р."
           value-title="Шлифовка лезвия - 150р." />
      <!--Упаковка набора элементов-->
      <pack type="data" key="dt7"
            elements="selected,editable,summ,allow-change-state,id3,name"/>
      <!--Упаковка набора элементов-->
      <pack type="nested" key="#services"
            elements="dt1,dt2,dt3,dt4,dt5,dt6,dt7"/>
      <!--Удаление переменных из контекста-->
      <clear keys="selected,editable,summ,allow-change-state,id3,name,dt1,
             dt2,dt3,dt4,dt5,dt6,dt7"/>
      <goto target="com_screen"/>
     </action>
    </actions>
   </screen>
   <screen type="Communal" decor="simple" title="Корзина" id="com_screen">
    <!--Отображение полей из объекта NestedData #services-->
    <fields key="#services"/>
    <actions>
     <!--Описание поведения сценария оплаты при нажатии "Далее"-->
     <action type="Next" title="ДАЛЕЕ">
      <!--Получение итоговой суммы группового платежа-->
      <sum key="summ" result-key="ssum" input-data="#services"
          filter="selected=1"/>
      <!--Передача значения числовой переменной в сумму платежа без учета
комиссии-->
      <set-sum from="ssum" type="units"/>
      <!--Удаление переменных из контекста-->
      <clear keys="ssum"/>
      <goto target="pay"/>
     </action>
     <!--Описание поведения сценария оплаты при нажатии "Назад"-→
     <goto-action type="Prev" title="НАЗАД" target="group_screen3"/>
     <!--Описание поведения сценария оплаты при нажатии "Выход"-->
     <goto-action type="Exit" title="ВЫХОД" target="exit"/>
     <action type="Edit">
      <!--Если #selected не равно null-->
      <if condition="is-not-null #selected">
```

```
      <then>
        <!--Распаковка набора элементов-->
        <unpack type="data" key="#selected"
                elements="editable,summ,allow-change-state,id3,name"/>
        <!--Объявление переменной selected-->
        <set key="selected" value="1" value-title="1"/>
        <!--Упаковка набора элементов-->
        <pack type="data" key="#selected" elements="selected,editable,summ,
            allow-change-state,id3,name"/>
        <!--Удаление переменных из контекста-->
        <clear keys="#selected,selected,editable,summ,
                       allow-change-state,id3,name"/>
        <goto target="com_screen"/>
      </then>
      <else>
        <!--Если #removed не равно null-->
        <if condition="is-not-null #removed">
          <then>
            <!--Распаковка набора элементов-->
            <unpack type="data" key="#removed" elements="name"/>
            <!--Отображение информационного диалога-->
            <dialog type="Info" title="Запрос на удаление"
               message="Вы действительно хотите удалить {0} из корзины?"
               timeout="10" default="cancel" mess-params="name">
              <actions>
                <!--Описание поведения сценария, если нажата кнопка "Да"-->
                <action type="okay" title="ДА">
                  <!--Объявление переменной selected-->
                  <set key="selected" value="0" value-title="0"/>
                  <!--Упаковка набора элементов-->
                  <pack type="data" key="#removed" elements="selected,name"/>
                  <!--Удаление переменных из контекста-->
                  <clear keys="name,selected,#removed"/>
                  <goto target="com_screen"/>
                </action>
                <action type="cancel" title="НЕТ">
                  <!--Удаление переменных из контекста-->
                  <clear keys="#removed,name"/>
                  <cancel/>
                </action>
              </actions>
            </dialog>
          </then>
```

```
          </if>
        </else>
      </if>
    </action>
   </actions>
  </screen>
</scenario>
```

Экран, соответствующий данному сценарию, приведен на рисунке 7.1.1.

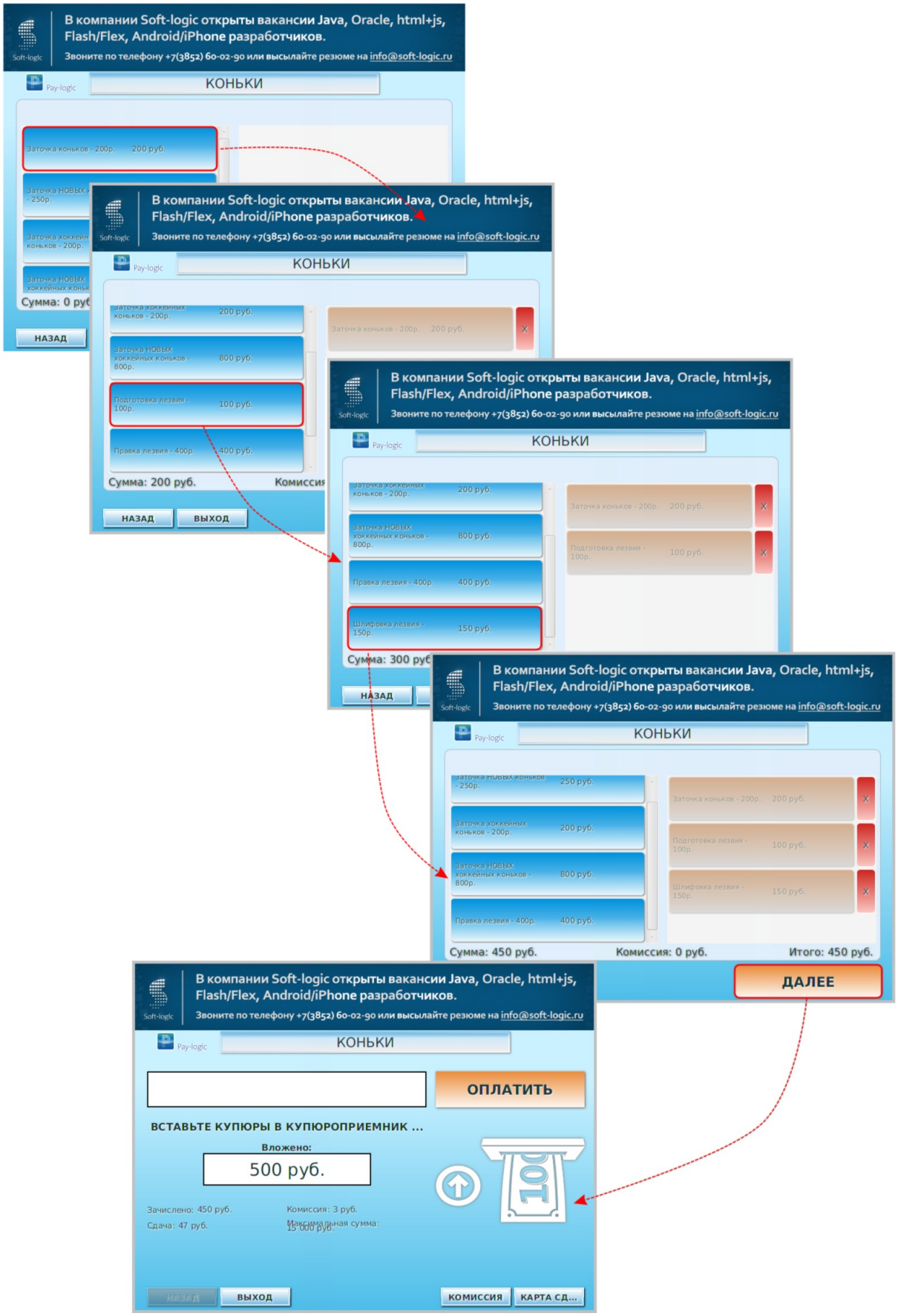


Рисунок 7.1.1 — Оплата с использованием корзины платежей

7.2 ПРОВЕДЕНИЕ ГРУППОВОГО ПЛАТЕЖА (<PAYMENT-PARAMS>)

Элемент, описываемый тегом `<payment-params>`, определяет особенности проведения группового платежа.

Структура:

```
<payment-params key=<notEmptyString15>
                single-check=[true|false]
                first-params-context=[true|false]
                distribution-type=[default|uniform|consistent]
                allow-return=[true|false]
                operation-order-items="[true|false]/>
```

Дочерние теги: отсутствуют.

Атрибуты приведены в таблице 7.2.1.

Таблица 7.2.1 — Атрибуты элемента <**payment-params**>

Атрибут	Описание	Обяз.
distribution-type	Используется для распределения остатка общей суммы платежей, указывается тип распределения сдачи. Возможные значения: • **default** — вложенная сумма распределяется по платежам согласно установленной в дочернем платеже сумме. Остаток выдаётся в виде сдачи; • **uniform** — вложенная сумма делится поровну на количество платежей. Если нельзя разделить всю сумму на равные платежи, то остаток от деления добавляется к сумме зачисленной первого платежа; • **consistent** — вложенная сумма	**Да**

Атрибут	Описание	Обяз.
	распределяется по платежам согласно установленной в дочернем платеже сумме. Остаток добавляется к сумме зачисленной первого платежа в пачке	
allow-return	Позволяет осуществлять возврат к предыдущим экранам ввода данных или в главное меню после того, как денежные средства уже внесены, сохраняя сумму для дальнейшей оплаты. Возможные значения: • **true** — возврат возможен. Значение по умолчанию; • **false** — возврат невозможен. Поддерживается ТПО 5 и 7. В случае, когда возврат возможен, пользователь может выбрать в модуле ввода: 1) либо выход из модуля с соответствующим формированием сдачи (выбором варианта); 2) либо вход в платежный модуль, но с новыми данными и прежним состоянием keeper-а (то есть с сохраненной суммой внесенных денежных средств)	Нет
first-params-context	Определяет порядок следования атрибутов платежа. Возможные значения: • **true** — приоритет у атрибутов платежа сценария, данные из корзины будут затерты; • **false** — приоритет у корзины атрибутов платежа, данные из сценария будут затерты	Нет
key	Указывается ключ объекта NestedData, который будет источником атрибутов групповой операции	Нет
operation-order-items	Настройка, предназначенная для соблюдения № 54-ФЗ, позволяет провести групповой платеж с	Нет

Атрибут	Описание	Обяз.
	фискализацией. Поддерживается на ТПО 7. • **true** — платеж будет проведен в новом формате, который учитывает необходимость фискализации; • **false** — проведение в старом формате. Для корректной работы настройки также необходимо, чтобы в теге <payment-params> был прописан параметр **key="<ключ объекта>"**. В результате проведения создается единая операция для всех услуг/товаров из корзины с их указанием в платеже. Объект NestedData в данном случае формируется иначе, подробнее см. в разделе 7.3	
single-check	Определяет, какой тип чека будет использован при печати. Возможные значения: • **true** — печатает общий чек на все платежи; • **false** — печатает отдельный чек на каждый платеж. Значение по умолчанию (если параметр не задан явно). Таким образом, если в сценарии: 1. Отсутствует <payment-params>, то печатается множество чеков. 2. Присутствует <payment-params>, но в нем не указана опция **single-check**, то печатается множество чеков. 3. Присутствует <payment-params> и указана опция **single-check="false"**, то печатается множество чеков. 4. Присутствует <payment-params> и указана опция **single-check="true"**, то печатается один чек	Нет

Атрибут	Описание	Обяз.
use-basket	Флаг позволяет использовать корзину для группового платежа — в качестве экрана корзины выступает экран **communal/basket**. Подробнее в разделе 4.6.2. Возможные значения: • **true** — платеж будет отправлен не сразу, а сначала добавится в корзину; • **false** — без использования корзины	Нет

Пример:

```
<payment-params key="#services" single-check="true"
                first-params-contex="false"
                distribution-type="consistent"/>
```

В данном примере источником атрибутов групповой операции является переменная **services**, печатается общий чек на весь набор платежей, сдача зачисляется на первый платеж из набора.

7.3 ПРОВЕДЕНИЕ ГРУППОВОГО ПЛАТЕЖА В СООТВЕТСТВИИ С № 54-ФЗ

Для соблюдения № 54-ФЗ используется настройка, позволяющая проводить групповой платеж с указанием таких атрибутов как НДС, наименование позиции и т. д., требующихся для фискализации. За это отвечает флаг **`operation-order-items`**, прописанный в теге `<payment-params>`. Чтобы настройка работала корректно, в `<payment-params>` также обязательно должен быть прописан параметр **`key`**.

При значении флага **`"true"`** объект NestedData будет формироваться иначе. В таблице ниже представлен обновленный набор параметров для элементов, входящих в NestedData.

Таблица 7.3.1 — Параметры элементов NestedData

Атрибут	Описание	Тип	Обяз.
`name`	Наименование позиции. Не более 128 символов. Если указано более 128 символов, то при фискализации платежа значение поля будет урезано до максимально допустимого значения	String	**Да**
`price`	Цена за единицу. Указывается в копейках с учетом НДС	Int	**Да**
`id`	Идентификатор позиции. Используется для сопоставления позиции с внешними учетными системами	String	Нет
`subject-type`	Идентификатор типа товара. Возможные значения приведены по ссылке. Если не задано, используется значение по умолчанию: **1** (товар)	Int	Нет
`tax-id`	Идентификатор налоговой ставки. Возможные значения приведены по ссылке.	Int	Нет

Атрибут	Описание	Тип	Обяз.
	Если не указано, то берется значение по умолчанию. Оно задается в бэк-офисе в свойствах агента на вкладке «Налоговый режим». Для этого нужно установить флаг «Использовать НДС» и указать его размер в параметре «Ставка НДС»		
count	Количество, указывается в граммах. Если товар штучный, необходимо указывать 1000, 2000 и т. д. В этом случае 1000 = 1 шт. Значение по умолчанию: **1000**	Int	Нет

Пример запроса:

```
<!-- Платные услуги ГБУЗ «Баксанская ЦРБ» -->
<scenario begin="init_screen_1" chnum-name="id1">
    <payment-params key="#services" single-check="true" distribution-
type="default" operation-order-items="true"/>

    <!-- Наполнение корзины 1 -->
    <screen type="Void" decor="simple" title="Наполнение корзины"
id="init_screen_1">
      <actions>
        <action type="Next" title="ДАЛЕЕ">
           <set key="selected" value="1"/>
           <set key="editable" value="0"/>
           <set key="allow-change-state" value="1"/>

           <!-- price указывается в копейках -->
           <set key="id" key-title="Идентификатор товара" value="1"/>
           <set key="price" key-title="Цена" value="10000" value-
title="10000"/>
           <set key="tax-id" key-title="Идентификатор налоговой ставки"
value="1" />
           <set key="count" key-title="Количество товара" value="1000"/>
           <set key="subject-type" key-title="Тип товара" value="1"/>
           <set key="id3" key-title="Услуга" value="Медицинский осмотр
при приеме на работу, профилактический медицинский осмотр (форма 086) -
100р" value-title="Медицинский осмотр при приеме на работу,
профилактический медицинский осмотр (форма 086)"/>
```

```
            <set key="name" key-title="Прайс" value="Медицинский осмотр
при приеме на работу, профилактический медицинский осмотр (форма 086) -
100р." value-title="Медицинский осмотр при приеме на работу,
профилактический медицинский осмотр (форма 086) - 100р."/>
            <!-- Упаковка элемента -->
            <pack type="data" key="dt1"
elements="selected,editable,id,price,tax-id,subject-type,count,allow-
change-state,id3,name"/>

             <set key="id" key-title="Идентификатор товара" value="2"/>
             <set key="price" key-title="Цена" value="15000" value-
title="15000"/>
            <set key="tax-id" key-title="Идентификатор налоговой ставки"
value="2" />
             <set key="count" key-title="Количество товара" value="1500"/>
             <set key="subject-type" key-title="Тип товара" value="2"/>
             <set key="id3" key-title="Услуга" value="1" value-title="Мед.
осмотр при приеме на раб-у, проф. мед. осмотр с вред. услов. труда"/>
             <set key="name" key-title="Прайс" value="Мед. осмотр при
приеме на раб-у, проф. мед. осмотр с вред. услов. труда - 150р." value-
title="Мед. осмотр при приеме на работу, проф-ий мед. осмотр с вред.
услов. труда - 150р."/>
             <pack type="data" key="dt2"
elements="selected,editable,id,price,tax-id,subject-type,count,allow-
change-state,id3,name"/>

             <set key="id" key-title="Идентификатор товара" value="3"/>
             <set key="price" key-title="Цена" value="20000" value-
title="20000"/>
             <set key="tax-id" key-title="Идентификатор налоговой ставки"
value="3" />
             <set key="count" key-title="Количество товара" value="2000"/>
             <set key="subject-type" key-title="Тип товара" value="3"/>
             <set key="id3" key-title="Услуга" value="1" value-title="Мед.
освид-ие на наличие мед-их против-й к управл. трансп. средством. Кат.В
(форма 003-В/у)"/>
             <set key="name" key-title="Прайс" value="Мед. освид-ие на
наличие мед-их против-й к управл. трансп. средством. Кат.В - 200р."
value-title="Мед. освид-ие на наличие мед-их против-й к управл. трансп.
средством. Кат.В - 200р."/>
             <pack type="data" key="dt3"
elements="selected,editable,id,price,tax-id,subject-type,count,allow-
change-state,id3,name"/>
```

```
            <!-- Упаковка набора элементов -->
            <pack type="nested" key="#services" elements="dt1,dt2,dt3"/>
            <clear keys="selected,editable,id,price,tax-id,subject-
type,count,allow-change-state,id3,name,dt1,dt2,dt3"/>
             <goto target="input_data"/>
          </action>
       </actions>
   </screen>


    <!-- Ввод данных клиента -->
    <screen type="group" title="Введите данные клиента" id="input_data">
        <fields>
            <text-field id="fio" title="ФИО" max-len="100" keyboard="any:
[ru,symb]:upper:true" default="Иванов Иван Иванович">
                <validator type="regex">
                    <rules>
                        <rule regex="^.{3,100}$"/>
                    </rules>
                </validator>
            </text-field>
            <text-field id="id1" title="Номер телефона" max-len="10"
keyboard="Digital" default="9130218122">
                <validator type="regex">
                    <rules>
                        <rule regex="^\d{10}$"/>
                    </rules>
                </validator>
                <formatter type="regex">
                    <rules default="8(***)***-**-**"/>
                </formatter>
            </text-field>
            <text-field id="contract" title="Номер договора" max-len="8"
keyboard="Digital" default="123456">
                <validator type="regex">
                    <rules>
                        <rule regex="^\d{1,8}$"/>
                    </rules>
                </validator>
            </text-field>
        </fields>
        <actions>
```

```
            <action type="Next" title="ДАЛЕЕ">
                <sum key="summ" result-key="ssum" input-data="#services"
filter="selected=1"/>
                <!--<set key="ssum" value="2400" value-title="0"/> -->
                <set-sum from="ssum" type="units"/>
                <clear keys="ssum"/>
                <goto target="pay"/>
            </action>
            <action type="Prev" title="НАЗАД">
                <goto target="com_screen"/>
            </action>
            <action type="Exit" title="ВЫХОД">
                <goto target="exit"/>
            </action>
        </actions>
    </screen>
</scenario>
```

7.4 ПОДГОТОВКА И ГРУППИРОВКА АТРИБУТОВ ПЛАТЕЖА (<COMMUNAL-PREPARE>)

В том случае, когда для оплаты используется корзина платежей, для передачи на сервер требуется сформировать атрибуты платежа на основе выбранных клиентом данных. Для подготовки атрибутов используется элемент, описываемый тегом `<communal-prepare>`.

Структура тега:

```
<communal-prepare input-data=<notEmptyString40>
                  filter=<notEmptyString40>>
  <elements>
    <input-element .../>
  </elements>
</communal-prepare>
```

Дочерние теги: `<elements>`.

Атрибуты приведены в таблице 7.4.1.

Таблица 7.4.1 — Атрибуты элемента `<communal-prepare>`

Атрибут	Описание	Обяз.
input-data	Атрибут, указывающий на переменную, которая содержит данные для обработки. В выборку попадают только те значения, которые удовлетворяют условию из атрибута **filter**	Нет
filter	Правило, согласно которому будут отбираться элементы из **input-data**	Нет

Пример:

```
<communal-prepare input-data="#services" filter="selected=1">
```

```
    ...
</communal-prepare>
```

В данном примере для формирования атрибутов платежа выбираются элементы из переменной **services** со значением параметра **«selected»**, равным **«1»**. Фильтр может быть и более сложным и содержать несколько критериев, например, **filter=«selected=1;tarrifcode=10»**.

Элемент, описываемый тегом <elements>, используется для группировки атрибутов платежа. Структура тега:

```
<elements>
      <input-element ... /> ...
</elements>
```

Дочерние теги: <input-element>.

Атрибуты: отсутствуют.

Тег <input-element> используется для описания правил создания атрибутов платежа из набора атрибутов платежа (NestedData). Структура тега:

```
<input-element key=<notEmptyString25>
               key-title=<notEmptyString25>
               value=<notEmptyString25>
               value-title=<notEmptyString25>/>
```

Атрибуты приведены в таблице 7.4.2.

Таблица 7.4.2 — Атрибуты элемента <input-element>

Атрибут	Описание	Обяз.
key	Указывается InputElement, содержащий ключ, получаемого атрибута	Да
key-title	Указывается InputElement, содержащий наименование, получаемого атрибута	Да
value	Указывается InputElement, содержащий значение, получаемого атрибута	Да

Атрибут	Описание	Обяз.
value-title	Указывается InputElement, содержащий значение получаемого атрибута, отображаемое на экране	Да

Пример:

```
<communal-prepare input-data="#counters">
   <elements>
     <input-element key="${name}" key-title="${name}"
                    value="${curr}" value-title="${curr}"/>
   </elements>
</communal-prepare>
```

В данном примере для формирования атрибутов платежа выбираются элементы из переменной **counters**. Ключ и наименование получаемого атрибута берутся из переменной **name**, значение и отображаемое значение — **curr**.

```
<communal-prepare input-data="#services" filter="selected=1">
   <elements>
     <input-element key="serv_${serviceid}" key-title="${name}"
                    value="${summ}" value-title="${summ}"/>
   </elements>
</communal-prepare>
```

В данном примере для формирования атрибутов платежа выбираются элементы из переменной **services** со значением параметра **«selected»**, равным **«1»**. Ключ, получаемого атрибута образуется путем конкатенации «serv_» и значения, получаемого из переменной **serviceid**, заголовок получаемого атрибута берется из значения атрибута **name**, значение и отображаемое значение — **summ**.

Элемент <communal-prepare> также может использоваться для индексации элементов корзины.

Пример:

```
<communal-prepare input-data="#seats-list" filter="selected=1">
  <elements>
    <input-element key="type.${real-data-count}"
                   key-title="Passenger type" value="${type}"
                   value-title="${type}" original-value="${type}"/>
```

```
  </elements>
</communal-prepare>
```

В данном случае по каждой выбранной услуге из корзины сформируется атрибут **`type.0`**, **`type.1`** и т.д.

7.5 УПАКОВКА НАБОРА ЭЛЕМЕНТОВ (<PACK>)

Тег <pack> упаковывает набор элементов **elements**. В зависимости от атрибута **type** в контексте создается объект с типом Nested или Data и ключом **key**. Полученный объект NestedData может быть использован для отображения на экранах селекторах терминалов или для формирования групповой операции (передача пакета платежей).

Структура тега:

```
<pack type=[data|nested]
      key=<notEmptyString25>
      elements=<keyListType>
      additive=[true|false]/>
```

Атрибуты приведены в таблице 7.5.1.

Таблица 7.5.1 — Атрибуты элемента <pack>

Атрибут	Описание	Обяз.
type	Тип передаваемого объекта. Возможные значения: • **data** — создается объект типа Data с набором элементов InputElement и, если объект с таким ключом уже существует, то новый элемент добавляется к этому набору. • **nested** — создается объект NestedData с набором элементов типа Data. Один элемент NestedData содержит Data элементы с разными InputElement	Да
key	Ключ упаковываемого объекта	Да
elements	Набор упаковываемых элементов (например, атрибуты платежей), переменные перечисляются через запятую без пробелов	Да

Атрибут	Описание	Обяз.
additive	Обрабатывается только если **type="nested"**. Возможные значения: 1. **false** — упаковка в nested новой data перезаписывает содержимое (текущее поведение). Значение по умолчанию. 2. **true** — упаковка в nested новой data добавляет обновляемые элементы к уже имеющимся. Если при этом элемента nested не существует, то он будет создан.	Нет
input-elements	По умолчанию имеет значение **false**. Обрабатывается только при **type="nested"**. Если параметр имеет значение **true**, то переменные в атрибуте **elements** обрабатываются как input-elementы и строка с данными input-elementами добавляется в новый или уже существующий элемент nested. **Пример:** `<pack type="nested" key="key" elements="id1,id2" additive="true" input-elements="true"/>`	Нет

Пример:

```
<pack type="data" key="data1" elements="id1,id2,fio"/>
<pack type="data" key="data2" elements="id1,sum"/>
<pack type="nested" key="ndata1" elements="data1,data2"/>
```

В набор атрибутов платежа **ndata1** включены элементы **data1** и **data2**. В элемент **data1** включены атрибуты **id1**, **id2**, **fio**, data2 — **id1**, **sum**.

Пример:

```
<action type="okay" title="ДА">
      <set key="selected" value="0" value-title="0"/>
```

```
        <pack type="data" key="#removed" elements="selected"/>
        <clear keys="#removed,name"/>
        <goto target="services_screen"/>
</action>
```

В данном примере в элемент **removed** типа Data включен атрибут **selected** со значением 0 и отображаемым значением 0.

7.6 РАСПАКОВКА ЭЛЕМЕНТОВ (<UNPACK>)

Тег `<unpack>` распаковывает набор элементов `<elements>`. Из объекта с ключом **`key`** берется набор элементов и вставляется в контекст. Используется для изменения объектов типа Data и NestedData.

Структура тега:

```
<unpack type=[nested|data]
        key=<keyType>
        elements=<keyListType>/>
```

Дочерние теги: отсутствуют.

Атрибуты аналогичны атрибутам элемента `<pack>` (раздел 7.5), за исключением того, что указываются атрибуты упаковываемого элемента и атрибут **`elements`** необязателен.

Пример:

```
<unpack type="nested" key="ndata1" elements="data1,data2"/>
<unpack type="data" key="data1" elements="id1,id2,fio"/>
<unpack type="data" key="data2" elements="id1,sum"/>
```

В данном примере распаковывается набор атрибутов платежа, включающий элемент **`data1`** и **`data2`.** А затем распаковываются элементы **`data1`** с атрибутами **`id1`**, **`id2`**, **`fio`** и **`data2`** с атрибутами **`id1`**, **`sum`**.

Теги `<pack>` и `<unpack>` используются совместно для изменения элементов InputElement — атрибутов платежа или Data — отдельного платежа из корзины платежей.

7.7 ИТОГОВАЯ СУММА (<SUM>)

Элемент, описываемый тегом <sum>, применяется к элементам корзины платежей (NestedData). Если представить nestedData таблицей, то <sum> обрабатывает столбцы таблицы (сумма значений InputElement). Позволяет получить значение итоговой суммы группового платежа.

Структура тега:

```
<sum key=<notEmptyString25>
     result-key=<notEmptyString25>
     input-data=<notEmptyString25>
     filter=<notEmptyString25>/>
```

Дочерние теги: отсутствуют.

Атрибуты описаны в таблице 7.7.1.

Таблица 7.7.1 — Атрибуты элемента <sum>

Атрибут	Описание	Обяз.
key	Указывает название суммируемых атрибутов (input-элементов) из набора данных (Data)	Да
result-key	Ключ переменной, в которую будет помещён результат суммирования, данный элемент будет помещён в контекст	Да
input-data	Атрибут, указывающий на переменную, которая содержит данные для обработки. В выборку попадают только те значения, которые удовлетворяют условию из атрибута **filter**	Да
filter	Правило, согласно которому будут отбираться	Нет

Атрибут	Описание	Обяз.
	элементы из **input-data**. Возможные варианты: • **filter="selected!="** — непустое значение InputElemet в столбце selected. • **filter="selected="** — пустое. • **filter="selected!=1"** — не равно 1. • **filter="selected=1"** — равно 1. • **filter="selected"** — пустое	

Пример:

```
<sum key="summ"
     result-key="ssum"
     input-data="services"
     filter="selected=1"/>
```

В данном примере для формирования итоговой суммы платежа выбираются элементы из переменной **services** со значением параметра **«selected»**, равным **«1»**. Суммируются значения атрибута **summ** и полученное значение присваивается переменной **ssum**.

7.8 НАБОР АТРИБУТОВ ДЛЯ ЭКРАНА ЖКХ

Для формирования данных, используемых на экране ЖКХ, с помощью корзины, используются следующие атрибуты:

1. **#id** — идентификатор для услуги.
2. **name** — задает наименование элемента.
3. **title** — заголовок элемента, если не указан используется **name**. Если указаны оба, используется первый встречающийся. Выводится в качестве надписи на кнопке с элементом.
4. **current** — текущее показание счетчиков по данному виду коммунальных услуг.
5. **previous** — предыдущее показание счетчиков по данному виду коммунальных услуг.
6. **tariff** — стоимость одной условной единицы потребленных услуг.
7. **consumption** — объем потребленных услуг за текущий период.
8. **summ** — сумма к оплате за данный вид коммунальных услуг .
9. **selected** — определяет выбран ли элемент. Может принимать значения 1 (элемент выбран) и 0 (элемент не выбран).

В рассматриваемом ниже примере на экране «VOID» сначала формируются данные: задаются типы услуг, для которых можно вводить показания: «Вода холодная», «Вода горячая», «Водоотведение», «Электроснабжение». Показания по услуге водоотведение определяются исходя из суммы показаний по услугам «Вода холодная», «Вода горячая».

Экран формирования данных пользователю не отображается, ему выводится экран выбора услуг с **id=«meter_screen»**.

Если клиент выбирает нажимает на ввод текущего показания по какой-либо из услуг, то он переходит к экрану ввода показаний приборов учета с **id=«input_sum1»**. После ввода показаний, осуществляется возврат на экран выбора услуг, с которого можно осуществить переход к экрану оплаты.

Пример:

```
<scenario xmlns="http://pay-logic.ru" begin="init_screen2">
 <!--Указание особенностей проведения группового платежа-->
 <payment-params key="#meters" distribution-type="default" single-
check="true" />
 <!--Экран формирования оффлайн данных-->
 <screen type="Void" decor="simple" title ="" id="init_screen2">
  <fields/>
  <actions>
   <action type="Next" title="Далее">
    <!--Объявление переменных-->
    <set key="#id" value="s1"/>
    <set key="name" value="Вода холодная" />
    <set key="current" value=""/>
    <set key="previous" value="153"/>
    <set key="tariff" value="2.5"/>
    <set key="consumption" value=""/>
    <set key="summ" value="" />
    <set key="selected" value="1"/>
    <!--Упаковка набора элементов-->
    <pack type="data" key="dat01" elements="#id,name,current,previous,
          tariff,consumption,summ,selected"/>
    <!--Объявление переменных-->
    <set key="#id" value="s2"/>
    <set key="name" value="Вода горячая" />
    <set key="current" value=""/>
    <set key="previous" value="85"/>
    <set key="tariff" value="3.5"/>
    <set key="consumption" value=""/>
    <set key="summ" value="" />
    <set key="selected" value="1"/>
    <!--Упаковка набора элементов-->
    <pack type="data" key="dat02" elements="#id,name,current,previous,
          tariff,consumption,summ,selected"/>
    <!--Объявление переменных-->
    <set key="#id" value="s3"/>
```

```
    <set key="name" value="Водоотведение" />
    <set key="current" value=""/>
    <set key="previous" value=""/>
    <set key="tariff" value="1.1"/>
    <set key="consumption" value=""/>
    <set key="summ" value="" />
    <set key="selected" value="1"/>
    <set key="#ref" value="s1,s2"/>
    <!--Упаковка набора элементов-->
    <pack type="data" key="dat03" elements="#id,name,current,previous,
          tariff,consumption,summ,selected,#ref"/>
    <!--Объявление переменных-->
    <set key="#id" value="s4"/>
    <set key="name" value="Электроэнергия" />
    <set key="current" value=""/>
    <set key="previous" value="11270"/>
    <set key="tariff" value="2.7"/>
    <set key="consumption" value=""/>
    <set key="summ" value="" />
    <set key="selected" value="1"/>
    <!--Упаковка набора элементов-->
    <pack type="data" key="dat04" elements="#id,name,current,previous,
          tariff,consumption,summ,selected"/>
    <pack type="nested" key="#meters"
          elements="dat01,dat02,dat03,dat04"/>
    <!--Удаление переменных из контекста-->
    <clear keys="#id,name,current,previous,tariff,consumption,
           summ,selected,#ref,dat01,dat02,dat03,dat04"/>
    <goto target="meter_screen"/>
   </action>
  </actions>
 </screen>
 <screen type="communal/meter" title="Выбор услуг" id="meter_screen">
  <!--Отображение полей из объекта NestedData #meters-->
  <fields key="#meters"/>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии "Далее"-->
   <action type="Next" title="Далее">
    <!--Получение итоговой суммы группового платежа-->
    <sum key="summ" result-key="ssum2" input-data="#meters"
         filter="selected=1" />
    <!--Передача значения числовой переменной в сумму платежа без учета
комиссии-->
```

```
    <set-sum from="ssum2" type="units"/>
     <goto target="pay"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии "Назад"-->
   <goto-action type="Prev" title="Назад" target="input"/>
   <!--Описание поведения сценария оплаты при нажатии "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
   <action type="Edit" title="РЕДАКТИРОВАТЬ">
    <!--Если #selected не равно null-->
    <if condition="is-not-null #selected">
     <then>
      <!--Распаковка набора элементов-->
      <unpack type="data" key="#selected"
              elements="current,previous,tariff,consumption,summ" />
       <goto target="input_sum1"/>
     </then>
     <else/>
    </if>
   </action>
  </actions>
 </screen>
 <screen type="Digital" title="Введите показания прибора учёта"
         id="input_sum1">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 5. Название поля
ввода: "Текущее показание"-->
   <text-field id="current" title="Текущее показание" max-len="5"
               keyboard="Digital">
    <validator type="regex">
     <rules>
      <rule regex="\d{1}.*$"/>
     </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого показания на
экране-->
    <formatter type="regex">
     <rules default="*** *** ***"/>
    </formatter>
     <!--Подсказка, отображается на экране-->
     <help> Введите показания прибора учёта </help>
   </text-field>
  </fields>
```

```
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
   <action type="Next" title="Далее">
    <math operation="current - previous" result-key="consumption"
          result-title="Потребление"/>
    <math operation="consumption * tariff" result-key="summ"
          result-title="К оплате"/>
    <!--Упаковка набора элементов-->
    <pack type="data" key="#selected"
          elements="current,previous,tariff,consumption,summ" />
    <!--Удаление переменных из контекста-->
    <clear keys="current,previous,tariff,consumption,summ"/>
    <goto target="meter_screen"/>
   </action>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
   <goto-action type="Prev" title="Назад" target="meter_screen"/>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
   <goto-action type="Exit" title="Выход" target="exit"/>
  </actions>
 </screen>
</scenario>
```

На рисунке 7.8.1 представлен переход между экранами, отражающий логику сценария.

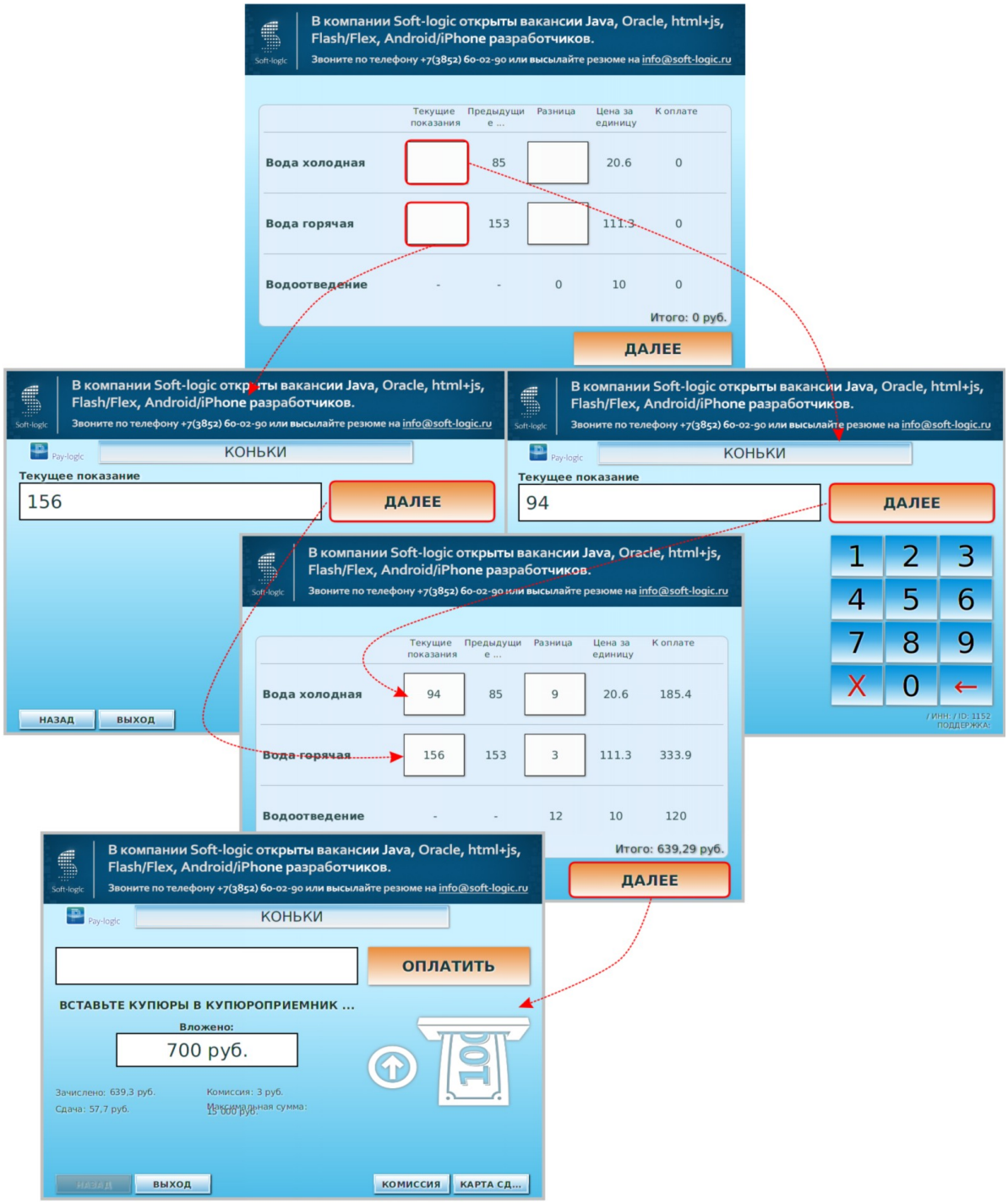


Рисунок 7.8.1 — Пример работы сценария

8 ТЕГИ ВАЛИДАТОРЫ

Теги-валидаторы (далее-валидатор) используются для проверок введенных данных, создаются на основе регулярных выражений.

Объявляется секция валидатора и задается тип валидатора с помощью атрибута **`type`** и в зависимости от типа имя валидатора, атрибут — **`name`.**

В процессинге используются следующие валидаторы: `<validator>` (раздел 5.14.1), `<data-formatter>` (раздел 5.14.2), `<formatter>` (раздел 5.14.3), `<modificator>` (раздел 5.14.4).

9 ПОЛУЧЕНИЕ ДАННЫХ ИЗ СПРАВОЧНИКА ТОЧКИ (<PREFILL-ITEMS>)

Элемент, описываемый тегом `<prefill-items>`, предназначен для добавления данных из справочника точки в контекст сценария. Также для заполнения полей данными из окружения служит элемент `<load>` (раздел 6.18).

Структура элемента:

```
<prefill-items>
      <item from=<notEmptyString25>
            to=<notEmptyString25>
            title=<notEmptyString40>/>
</prefill-items>
```

Дочерние элементы: `<item>`.

Атрибуты элемента `<item>` приведены в таблице 9.1.

Таблица 9.1 — Атрибуты элемента `<item>`

Атрибут	Описание	Обяз.
`from`	Указывает, какой элемент берется из справочника	Да
`to`	Указывает, какому атрибуту платежа присваивается значение	Да
`title`	Указывает заголовок для полученного атрибута. Если атрибут не указан, то будет использовано название по умолчанию	Нет

Пример:

```
<prefill-items>
      <item from="point.city" to="city" title="Город"/>
</prefill-items>
```

Название города из справочника присваивается атрибуту **`city`**.

9.1 ПОЛУЧЕНИЕ ДАННЫХ ДЛЯ ТПО

9.1.1 АТРИБУТЫ АГЕНТА

ТПО версий 5, 7 поддерживает запрос следующих атрибутов агента:

1. **`dealer.id`** — идентификатор агента.
2. **`dealer.name`** — наименование агента.
3. **`dealer.inn`** — ИНН агента.
4. **`dealer.address`** — адрес агента.
5. **`dealer.support`** — контакты поддержки.
6. **`dealer.supportHours`** — время работы поддержки.
7. **`dealer.contractNo`** — номер договора.
8. **`dealer.contractDate`** — дата договора.
9. **`dealer.account`** — расчетный счет.
10. **`dealer.bank`** — банк.
11. **`dealer.bik`** — БИК банка.
12. **`dealer.kpp`** — КПП агента.
13. **`dealer.okato`** — ОКАТО агента.
14. **`dealer.ks`** — корреспондентский счет.

9.1.2 АТРИБУТЫ СЕРВИСА

Данные о сервисе извлекаются из справочника ТПО 5 при помощи конструкции **`menuIt.service.<название свойства>`**, из ТПО 7 — при помощи **`service.<название свойства>`**.

Для ТПО 5 возможно получить следующие данные о сервисе:

1. **`menuItem.service.id`** — ID сервиса.
2. **`menuItem.service.code`** — код сервиса.
3. **`menuItem.service.barcodePrefix`** — префикс штрих-кода.
4. **`menuItem.service.name`** — сервис.
5. **`menuItem.service.printName`** — название для печати.
6. **`menuItem.service.altName`** — альтернативное название.
7. **`menuItem.service.invitation`** — текст приглашения.
8. **`menuItem.service.minSum`** — минимальная сумма платежа.
9. **`menuItem.service.maxSum`** — максимальная сумма платежа.
10. **`menuItem.service.maxInputSum`** — максимальная вложенная сумма платежа.
11. **`menuItem.service.divideSum`** — сумма разбивки платежа.
12. **`menuItem.service.verifyType`** — тип верификации.
13. **`menuItem.service.processingType`** — тип проведения.
14. **`menuItem.service.minComm`** — минимальная комиссия.

15. **menuItem.service.paymentSources** — источники оплаты.

16. **menuItem.service.getMaxComm()** — комиссия, такие коды связаны с особенностями velocity (отсутствует стандартный геттер для поля). Есть альтернативное название "commission".

17. **menuItem.service.getOutCurrency()** — валюта провайдера, альтернативное название "out-currency".

18. **menuItem.service.getInCurrency()** — валюта сервиса, альтернативное название "in-currency".

Для ТПО 7 доступны те же данные, но в синтаксисе не указывается **menuItem.**

9.1.3 АТРИБУТЫ ТОЧКИ

ТПО версий 5, 7 поддерживает запрос следующих атрибутов точки:

1. **point.id** — номер точки.

2. **point.code** — код точки.

3. **point.name** — название точки.

4. **point.address** — адрес точки.

5. **point.city** — город, в котором находится точка.

6. **point.cityCode** — телефонный код города, в котором находится точка.

7. **point.area** — регион, в котором находится точка.

8. **point.params** — параметры точки.

9. **point.incassLimit** — лимит точки до следующей инкассации.

10. **point.pointType** — тип точки.

9.1.4 АТРИБУТЫ ПРОВАЙДЕРА

Атрибуты провайдера выводятся с помощью элемента `<load>` (раздел 6.18). Атрибуты провайдера можно получить из свойств сервиса (с вкладки «Основное») или из свойств провайдера проведения платежей. Данные будут отличаться в ситуациях, когда провайдером проведения и провайдером, предоставляющим услугу, являются разные поставщики. Например, когда провайдером проведения является агрегатор платежей — в этом случае данные поставщика услуги будут указаны в свойствах сервиса, а данные провайдера проведения, т. е. агрегатора, — в свойствах провайдера.

Для получения атрибутов провайдера используйте следующие конструкции:

1. Для вывода данных из свойств сервиса:

   1) **`service.provider.name`** — наименование получателя.

   2) **`service.provider.inn`** — ИНН получателя.

   3) **`service.provider.support`** — контакты поддержки получателя.

   Обратите внимание, что если в свойствах сервиса не задана соответствующая информация о провайдере, то вернется пустое поле.

2. Для вывода данных из свойств провайдера проведения применяются **`service.operator.*`** и **`service.provider.*`**:

   1) **`service.operator.name`** — наименование провайдера, пример:

   ```
   <load from="service.operator.name" key="id2" title="Имя провайдера"/>
   ```

   2) **`service.operator.inn`** — ИНН провайдера.

   3) **`service.operator.support`** — контакты поддержки.

   4) **`service.provider.id`** — идентификатор провайдера.

   5) **`service.provider.address`** — адрес провайдера.

6) **service.provider.supportHours** — время работы поддержки.

7) **service.provider.contractNo** — номер договора.

8) **service.provider.contractDate** — дата договора.

9) **service.provider.account** — расчетный счет.

10) **service.provider.bank** — банк.

11) **service.provider.bik** — БИК банка.

12) **service.provider.kpp** — КПП провайдера проведения.

13) **service.provider.okato** — ОКАТО провайдера проведения.

14) **service.provider.ks** — корреспондентский счет.

Конструкция **service.operator.*** обрабатывается ТПО 5 и 7, конструкция **service.provider.*** — только ТПО 5.

9.1.5 АТРИБУТЫ ПЛАТЕЖА

Для ТПО 5, 7 до версий x.121 идентификатор зарезервированной до оплаты операции доступен при помощи:

1. **operation.id** — идентификатор операции, пример:

```
<load from="operation.id" key="operation-id" title="ID операции" />
```

С версии x.121 конструкция `load` для вывода зарезервированного ID не нужна. Идентификатор сохраняется либо:

1. По умолчанию под ключом **operation-id** — в случае, когда в кабинете в параметрах точки установлен параметр **«Получение ID операции перед**

созданием платежа на терминале», а для advanced-запроса дополнительно не задан атрибут **`id-operation-param-name`**.

2. В произвольно заданном ключе из атрибута **`id-operation-param-name`**, если он настроен в advanced-запросе. Подробнее см. в разделе 6.6.

9.1.6 ПРОИЗВОЛЬНЫЕ СВОЙСТВА ОБЪЕКТОВ

Поддерживается в ТПО 5, 7. Чтобы получить произвольные свойства объекта, используйте следующие конструкции:

1. Для агента:

```
dealer.properties.<код произвольного свойства>
```

2. Для точки:

```
point.attributes.<код произвольного свойства>
```

3. Для сервиса:

```
service.properties.<код произвольного свойства>
```

4. Для провайдера проведения платежа:

```
service.provider.properties.<код произвольного свойства>
```

Чтобы указать название произвольного свойства, используйте **`title`**:

```
<load from="service.properties.country_code_id" title="Код страны
сервиса" key="id3"/>
```

9.2 АТРИБУТЫ ДЛЯ ЭЛЕКТРОННОГО КОШЕЛЬКА

Используются следующие атрибуты:

1. **`user.address`** — адрес пользователя.
2. **`user.email`** — адрес электронной почты пользователя.

> **Внимание!**
> Для отправки произвольного свойства на точку в настройках произвольного свойства в разделе «Справочники — Система — Типы свойств объектов» установите флаг **«Включать в справочники точки (отправка на точку)»**.

3. **`user.first-name`** — имя пользователя.
4. **`user.last-name`** — фамилия пользователя.
5. **`user.login`** — идентификатор учетной записи пользователя (логин).
6. **`user.middle-name`** — отчество пользователя.
7. **`user.passport.number`** — номер паспорта пользователя.
8. **`user.passport.series`** — серия паспорта пользователя.

9.3 ПРИМЕРЫ СЦЕНАРИЕВ

Пример сценария для вывода произвольных свойств (5 (7im) версия ТПО):

```
<?xml version="1.0" encoding="UTF-8"?>
<scenario xmlns="http://pay-logic.ru" begin="screen_1">
  <!--Добавление данных из справочников точки в контекст сценария-->
 <prefill-items>
  <item from="dealer.properties.recipient_name" to="d1"/>
  <item from="point.attributes.fp" to="d2"/>
 </prefill-items>
 <!--Экран ввода данных с несколькими полями-->
 <screen type="group" id="screen_1" title="тел">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 10. Название поля
ввода: "Лицевой счет"-->
   <text-field id="id1" title="Лицевой счет" max-len="10" keyboard="Digital"
               message="Введите лицевой счет">
    <!--Регулярное выражение для валидации вводимого лицевого счета-->
    <validator type="regex">
     <rules>
      <rule regex="^\d{1,10}$"/>
     </rules>
    </validator>
   </text-field>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 10. Название поля
ввода: "Номер телефона"-->
   <text-field id="phone" title="Номер телефона" max-len="10"
               keyboard="Digital" message="Введите номер телефона">
    <!--Регулярное выражение для валидации вводимого номера-->
    <validator type="regex">
     <rules>
      <rule regex="^\d{1,10}$"/>
     </rules>
    </validator>
   </text-field>
  </fields>
```

```
  <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="Далее">
      <goto target="confirm"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-→
    <action type="Prev" title="Назад">
      <goto target="previous"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <action type="Exit" title="Выход">
      <goto target="exit"/>
    </action>
  </actions>
 </screen>
 <screen type="confirm" title="Проверка данных" id="confirm">
  <!--Список атрибутов, отображаемых на экране подтверждения-->
  <fields list="d1,d2"/>
  <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="Далее">
      <goto target="pay"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <action type="Prev" title="Назад">
      <goto target="screen_1"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <action type="Exit" title="Выход">
      <goto target="exit"/>
    </action>
  </actions>
 </screen>
</scenario>
```

Экран **confirm**, отображающий произвольные свойства объектов, подгруженные в сценарий, приведен на рисунке 9.3.1.

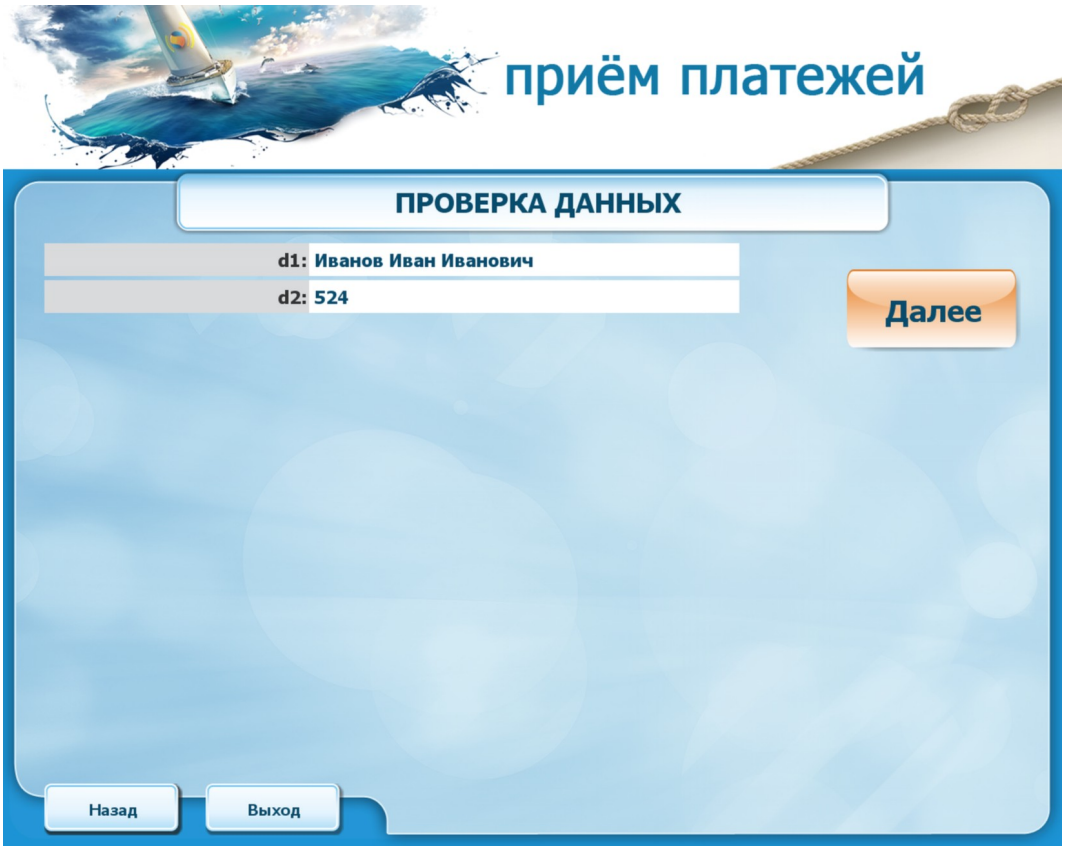


Рисунок 9.3.1 — Экран **confirm**, отображающий произвольные свойства объектов, подгруженные в сценарий

Для ТПО 5 и РМА 6 в **<prefill-items>** добавлена обработка атрибута **flags**.

Пример:

```
<scenario begin="1screen">
<!--Добавление данных из справочников точки в контекст сценария-->
  <prefill-items>
    <item from="paymentSystem.name" to="id1"
          key-title-id="Платежная система"/>
    <item from="point.code" to="id2" key-title-id="Код точки"
          flags="HIDE_ON_CONFIRM"/>
```

```
   <item from="point.city" to="id3" key-title-id="Город"/>
   <item from="menuItem.service.properties(account1)" to="id4"
         key-title-id="Расчетный счет"/>
  </prefill-items>
  <!--Создание экрана ввода данных с несколькими полями-->
  <screen type="group" title ="Введите данные" id="1screen">
   <fields>
    <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и бувенная клавиатура, язык ввода - русский без возможности
переключения языка, максимальное количество вводимых символов 60. Название
поля ввода: "ФИО плательщика". Атрибут отправляется в маскированном виде-->
    <text-field id="fio" title="ФИО плательщика" max-len="60"
                keyboard="any:[ru,symb]:upper:true"
flags="FLAG_MASK_IN_LOGS">
     <!--Регулярное выражение для валидации вводимых данных-->
     <validator type="regex">
      <rules>
       <rule regex="^[a-яА-Яёё-]{2,20}[\s]{1}[a-яА-Яёё-]{2,20}[\s]{1,3}
                    [a-яА-ЯЁё-]{2,20}([\s]{1}[a-яА-ЯЁё-]{2,20})?$"/>
      </rules>
     </validator>
     <!--Подсказка, отображается на экране-->
     <help> <![CDATA[Введите ФИО плательщика]]> </help>
    </text-field>
    <!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и бувенная клавиатура, язык ввода - русский без возможности
переключения языка, максимальное количество вводимых символов 254. Название
поля ввода: "Адрес плательщика"-->
    <text-field id="address" title="Адрес плательщика" max-len="254"
                     keyboard="any:[ru,symb]:upper:true">
     <!--Регулярное выражение для валидации вводимых данных-->
     <validator type="regex">
      <rules> <rule regex="^.{3,254}$"/> </rules>
     </validator>
     <!--Подсказка, отображается на экране-->
     <help> <![CDATA[Введите ФИО плательщика]]> </help>
    </text-field>
    <!--Создание поля выбора типа оплаты-->
    <selector-field id="type" title="Тип оплаты">
     <items type="static">
      <item value="21" title="Паспорт РФ"/>
      <item value="22" title="Заграничный паспорт"/>
      <item value="31" title="Паспорт иностранного гражданина"/>
```

```
      </items>
    </selector-field>
  </fields>
  <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="Далее">
      <goto  target="confirm"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <action type="Prev" title="Назад">
      <goto target="previous"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Exit" title="Далее">
      <goto target="exit"/>
    </action>
  </actions>
 </screen>
 <!--Создание экрана подтверждения-->
 <screen type="confirm" title ="Подтверждение данных платежа" id="confirm">
  <!--Список атрибутов, отображаемых на экране подтверждения-->
  <fields list="id1,id2,id3,id4"/>
  <actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
    <action type="Next" title="Далее">
      <goto target="pay"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
    <action type="Prev" title="Назад">
      <goto target="1screen"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <action type="Exit" title="Выход">
      <goto target="exit"/>
    </action>
  </actions>
 </screen>
</scenario>
```

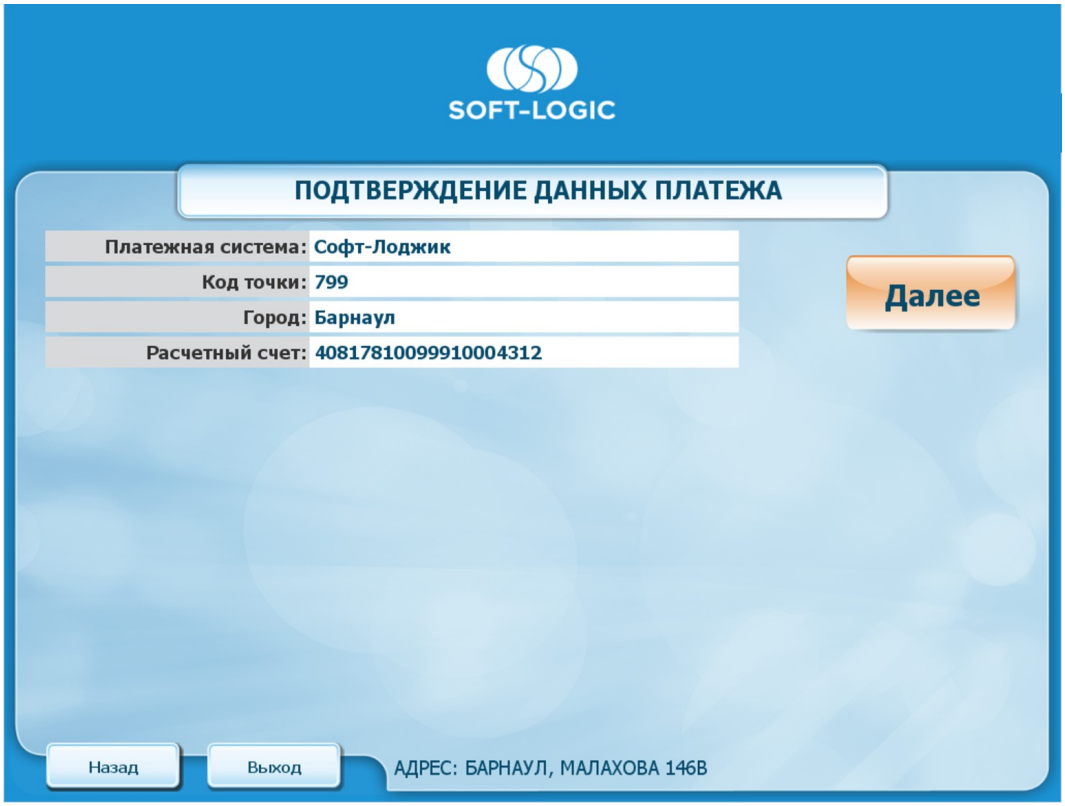


Рисунок 9.3.2 — Пример экрана **`confirm`**, если для атрибута «Код точки» в `<prefill-items>` не задан `flags=«HIDE_ON_CONFIRM»`

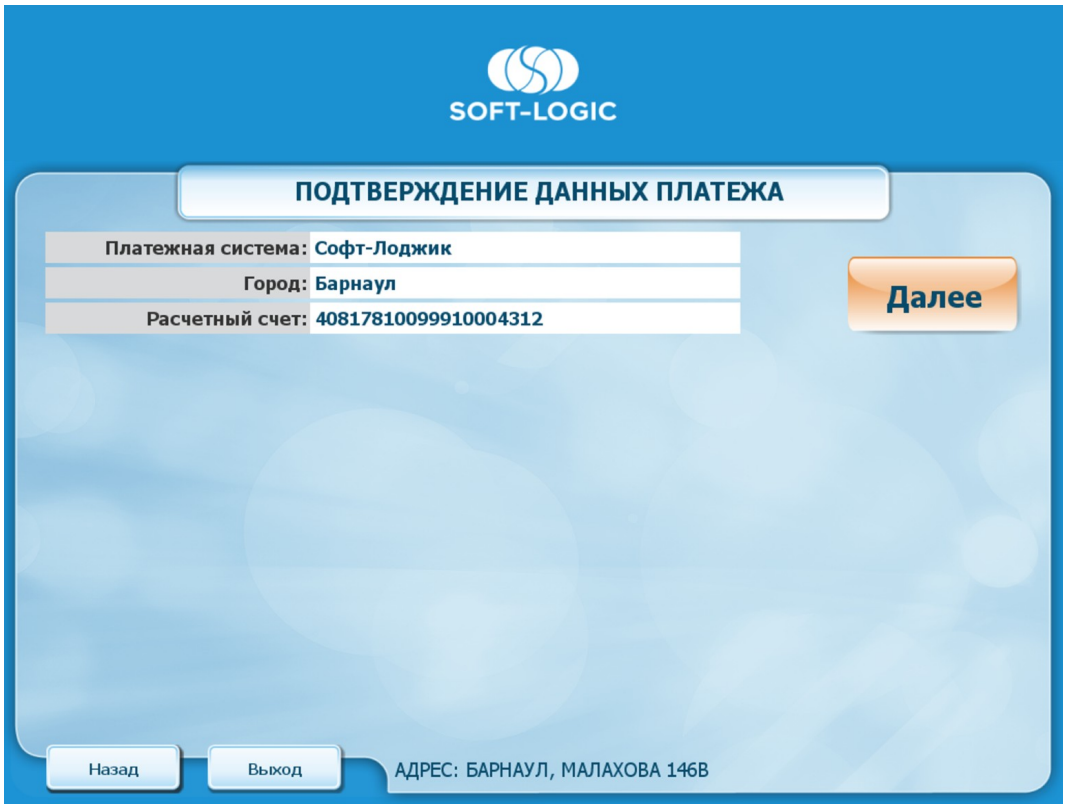


Рисунок 9.3.3 — Пример экрана **`confirm`**, если для атрибута «Код точки» в `<prefill-items>` задан `flags=«HIDE_ON_CONFIRM»`

В **`<prefill-items>`** добавлена обработка атрибута **`point.pointType`**. Данный атрибут позволяет передавать тип точки с сервера в составе справочника точки. Возможные типы точек и соответствующие им значения кодов приведены в таблице 9.3.1.

Таблица 9.3.1 — Значения атрибута **point.pointType**

Название точки	Тип точки	Код
Терминал	APPARAT	0
Офис	OFFICE	1
Мобильная точка	MOBILE_POINT	2
Шлюз	GATE	3
РМА	RMA	4
Шлюз по сертификату	GATE2	5
Шлюз по протоколу 2	GATE_CYBER	6
Касса с выдачей	CASH	7
POS-терминал	POS	8
Точка ВПС	VPS_POINT	9
Обменник	EXCHANGER	10
Мобильный телефон	CELLPHONE	11
Web Kiosk	WEB_KIOSK	12

Пример конструкции для получения значения типа точки в сценарии:

```
<item from="point.pointType" to="id2" title="Тип точки"/>
```

Пример конструкции для отображения полученного значения типа точки в сценарии:

```
<screen type="confirm" title="Подтверждение данных платежа" id="confirm">
<!--Список атрибутов, отображаемых на экране подтверждения-->
<fields list="id2"/>
</screen>
```

Пример сценария с использованием данной конструкции:

```
<scenario begin="1screen">
<!--Добавление данных из справочников точки в контекст сценария-->
<prefill-items>
<item from="paymentSystem.name" to="id1"
```

```
key-title-id="Платежная система"/>
<item from="point.pointType" to="id2" key-title-id="Тип точки"
flags="HIDE_ON_CONFIRM"/>
<item from="point.city" to="id3" key-title-id="Город"/>
<item from="menuItem.service.properties(account1)" to="id4"
key-title-id="Расчетный счет"/>
</prefill-items>
<!--Создание экрана ввода данных с несколькими полями-->
<screen type="group" title ="Введите данные" id="1screen">
<fields>
<!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и бувенная клавиатура, язык ввода – русский без возможности
переключения языка, максимальное количество вводимых символов 60.
Название
поля ввода: "ФИО плательщика". Атрибут отправляется в маскированном
виде-->
<text-field id="fio" title="ФИО плательщика" max-len="60"
 keyboard="any:[ru,symb]:upper:true" flags="FLAG_MASK_IN_LOGS">
<!--Регулярное выражение для валидации вводимых данных-->
<validator type="regex">
<rules>
<rule regex=".*"/>
</rules>
</validator>
<!--Подсказка, отображается на экране-->
<help> <![CDATA[Введите ФИО плательщика]]> </help>
</text-field>
<!--Создание поля ввода, со следующими параметрами: активна цифровая,
символьная и бувенная клавиатура, язык ввода – русский без возможности
переключения языка, максимальное количество вводимых символов 254.
Название
поля ввода: "Адрес плательщика"-->
<text-field id="address" title="Адрес плательщика" max-len="254"
 keyboard="any:[ru,symb]:upper:true">
<!--Регулярное выражение для валидации вводимых данных-->
<validator type="regex">
<rules> <rule regex="^.{3,254}$"/> </rules>
</validator>
<!--Подсказка, отображается на экране-->
<help> <![CDATA[Введите ФИО плательщика]]> </help>
</text-field>
<!--Создание поля выбора типа оплаты-->
<selector-field id="type" title="Тип оплаты">
```

```
<items type="static">
<item value="21" title="Паспорт РФ"/>
<item value="22" title="Заграничный паспорт"/>
<item value="31" title="Паспорт иностранного гражданина"/>
</items>
</selector-field>
</fields>
<actions>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
<action type="Next" title="Далее">
<goto  target="confirm"/>
</action>
<!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
<action type="Prev" title="Назад">
<goto target="previous"/>
</action>
<!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
<action type="Exit" title="Далее">
<goto target="exit"/>
</action>
</actions>
</screen>
<!--Создание экрана подтверждения-->
<screen type="confirm" title ="Подтверждение данных платежа"
id="confirm">
<!--Список атрибутов, отображаемых на экране подтверждения-->
<fields list="id1,id2,id3,id4"/>
<actions>
<!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
<action type="Next" title="Далее">
<goto target="pay"/>
</action>
<!--Описание поведения сценария оплаты при нажатии кнопки "Назад"-->
<action type="Prev" title="Назад">
<goto target="1screen"/>
</action>
<!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
<action type="Exit" title="Выход">
<goto target="exit"/>
</action>
</actions>
</screen>
</scenario>
```

Пример отображения полученных данных о типе точки представлен на рисунке 9.3.4.

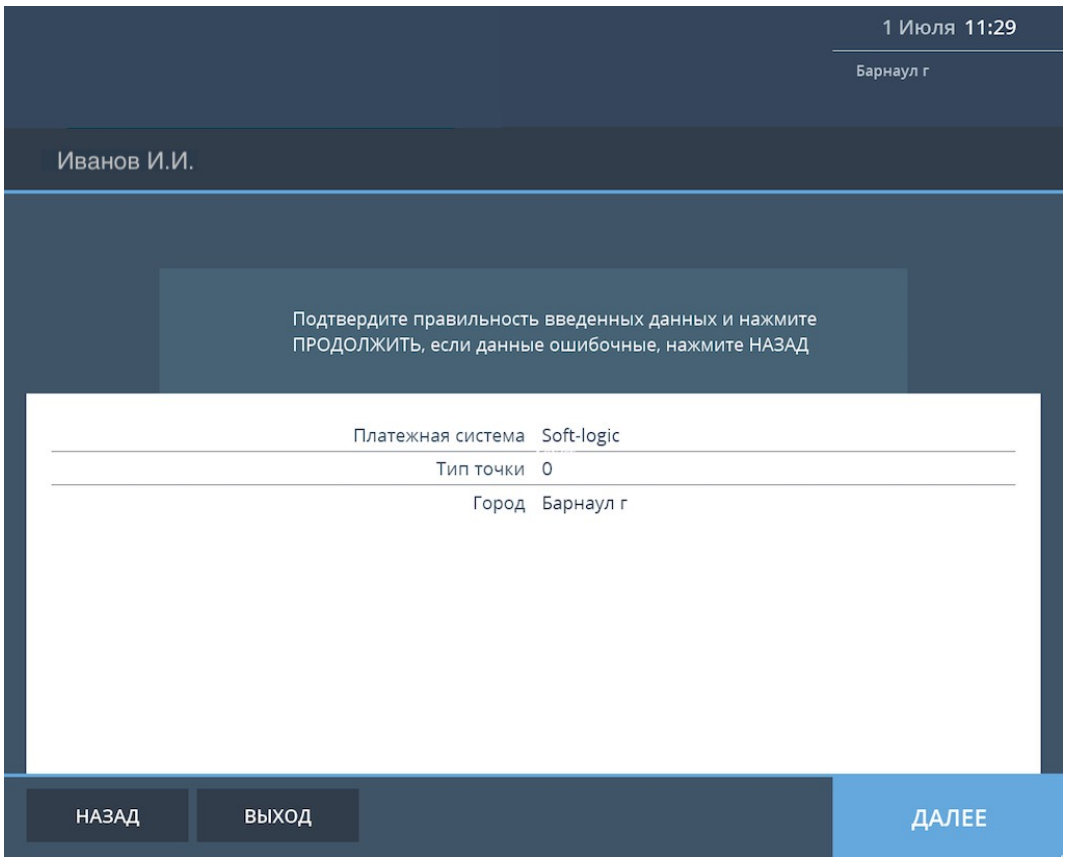


Рисунок 9.3.4 — Пример вывода значения типа точки

10 СПЕЦИАЛЬНЫЕ ВОЗМОЖНОСТИ

10.1 НАСТРОЙКА ГОЛОСОВОГО СОПРОВОЖДЕНИЯ

В ТПО 5 и 7 версий поддерживается возможность голосового сопровождения шагов оплаты.

Воспроизводимые файлы должны располагаться в каталоге ***<корень ТПО>/res/sound/*** или в каталоге ***<корень ТПО>/res/sound/<код локали>/*** в 7 версии ТПО и ***<корень ТПО>/sound/***, ***<корень ТПО>/sound/<код локали>/*** в 5 версии ТПО. Первоначально поиск файла осуществляется в каталоге ***<корень ТПО>/res/sound/<код локали>/*** в 7 версии ТПО и ***<корень ТПО>/sound/<код локали>/*** в 5 версии ТПО. В случае, если он оказывается не найден, то поиск осуществляется в папке ***sound*** по умолчанию.

Поддерживаются два типа воспроизведения файлов:

1. С начала до конца, не прерываются другими (название таких файлов должно начинается с символа нижнего подчеркивания «_»).

2. С прерыванием (воспроизведение файла прерывает воспроизведение текущего).

Поддерживаются следующие стандартные позиции звуковых файлов:

1. *_click.mp3* — на нажатие любой кнопки.

2. *hello.mp3* — при выходе в меню или перерисовке меню (по умолчанию проигрывается 2 минуты).

3. *operator.mp3* — при выборе любой группы в меню.

4. *pay_data.mp3* — при открытии экрана оплаты.

5. *complete.mp3* — на финальном диалоге.

Возможно воспроизводить звуки при открытии экранов ввода и при выполнении того или иного действия сценария.

Для воспроизведения файла при показе экране необходимо указать его в атрибуте **sound** элемента <screen>.

Пример:

```
<screen
       type="Selector"
       decor="currency"
       title="SELECTATI VALOAREA DE REINCARCARE"
       id="first_screen"
       sound="first_screen.mp3">
```

Для озвучки шагов сценария необходимо указать имя файла в атрибуте **file** элемента <play> действия <action>.

Пример:

```
<action ...>
       <play file="file.mp3"/>
</action>
```

10.2 ИСПОЛЬЗОВАНИЕ ПЕРЕМЕННОЙ, ОБОЗНАЧАЮЩЕЙ КОРНЕВОЙ КАТАЛОГ, ПРИ УКАЗАНИИ ПУТЕЙ В СЦЕНАРИЯХ

При необходимости использования ссылок на внешние файлы в сценариях возможно использовать переменную **$terminal_home**, принимающую значение корневого каталога ТПО. Примером использования может служить вывод изображения в разделе с подсказкой.

Пример (5 версия ТПО):

```
<?xml version="1.0" encoding="UTF-8"?>
<scenario xmlns="http://pay-logic.ru" begin="input_sum1">
 <!--Экран ввода числовой и текстовой информации-->
 <screen type="Digital" title="Введите показания прибора учёта"
         id="input_sum1" decor="info">
  <fields>
   <!--Создание поля ввода, со следующими параметрами: активна цифровая
клавиатура, максимальное количество вводимых символов 5. Название поля
ввода: "Текущее показание"-->
   <text-field id="current" title="Текущее показание" max-len="5"
               keyboard="Digital">
    <!--Регулярное выражение для валидации вводимых данных-->
    <validator type="regex">
     <rules> <rule regex="\d{1}.*$"/> </rules>
    </validator>
    <!--Регулярное выражение для форматирования вводимого показания на
экране-->
    <formatter type="regex"> <rules default="*** *** ***"/></formatter>
    <!--Подсказка, отображается на экране-->
    <help>
     <![CDATA[<html>Введите показания приборов учета
        <img src="$terminal_home/zk_bliznai_dacha.png"/>]]>
    </help>
   </text-field>
  </fields>
  <actions>
   <!--Описание поведения сценария оплаты при нажатии кнопки "Далее"-->
```

```
    <action type="Next" title="Далее">
      <goto target="pay"/>
    </action>
    <!--Описание поведения сценария оплаты при нажатии кнопки "Выход"-->
    <action type="Exit" title="Выход">
      <goto target="exit"/>
    </action>
</actions>
</screen>
</scenario>
```

Экран, соответствующий примеру, приведен на рисунке 10.2.1.

Рисунок 10.2.1 — Пример вывода изображения с использованием переменной, обозначающей корневой каталог ТПО

11 ОТЛИЧИЯ ТПО 5 И 7 ВЕРСИИ

Существуют следующие отличия между 5 и 7 версиями ТПО:

1. В системе обновлений сценарии располагаются в каталогах ***<корень ТПО>/resources/scenario/*** для 5 версии ТПО и ***<корень ТПО>/res/module/input/advanced/*** для 7 версии ТПО.

2. Ресурсные файлы с текстовками в ТПО 7 версии расположены в ***<корень ТПО>/res/module/i18n/***, в ТПО 5 версии — ***<корень ТПО>/resources/i18n/***.

3. Шаблоны чеков должны располагаться в каталоге ***<корень ТПО>/templates/default/custom/*** в 5 версии ТПО или ***<корень ТПО>/res/templates/default/custom/*** в 7 версии ТПО.

4. Для языка сценариев в 7 версии ТПО доступен тип экранов с навигацией. В 5 версии недоступен.

5. Атрибут **type** элемента `<init>` тега `<items>` с типом **calendar** может принимать значения Day, Month, Year (день, месяц, год) — в 7 версии ТПО. В 5 версии ТПО не реализовано.

6. Тип **local-capacity** сложного валидатора `<complex-validator>` в 7 версии ТПО не используется. При использовании `<complex-validator>` с типом **type=«bank-account»** в 5 версии атрибуты пишутся следующим образом: **bik-key**, **bank-key**, **account-key** типы действий: `<action type="bik-error">`, `<action type="account-error">`.

7. Воспроизведение звуков при открытии экранов ввода реализовано только в 5 версии ТПО.

12 НЕКОТОРЫЕ ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

Задача 1. Необходимо начать сценарий с диалога.

Решение: использовать тип экрана void.

```
<scenario xmlns="http://pay-logic.ru" begin="first_screen">
  <screen type="void" decor="simple" title="Enter detail. Page 1."
          id="first_screen">
     <dialog type="Info" title="Внимание"
             message="Disclaimer: This machine only accepts money bills
             and does not dispense change. If your money is more than the
             amount due, this machine will issue a change card as receipt
             which you can use to purchase Mobile or Internet load being
             sold in this machine."
             timeout="10" default="cancel">
       <actions>
         <goto-action type="okay" title="OK" target="first_screen"/>
         <goto-action type="cancel" title="CANCEL" target="exit"/>
       </actions>
     </dialog>
  </screen>
  ...
</scenario>
```

Задача 2. Необходимо отображать только экран ввода денег. Может использоваться в случаях, когда необходимо продать конкретный товар с конкретной ценой.

```
<scenario xmlns="http://pay-logic.ru" begin="first_screen">
  <!--Создание экрана формирования данных-->
  <screen type="Void" decor="button" title ="" id="first_screen">
    <fields/>
      <actions>
        <!--Описание поведения сценария оплаты при нажатии кнопки
"Next"-->
        <action type="Next" title="NEXT">
          <!--Если #back не равно null-->
          <if condition="is-not-null #back">
            <then>
              <!--Переход на предыдущий экран-->
              <goto target="previous"/>
```

```
            </then>
            <else>
              <!--Объявление переменной #back-→
              <set key="#back" value="1"/>
              <!--Переход на экран оплаты-->
              <goto target="pay"/>
            </else>
          </if>
        </action>
      </actions>
    </screen>
  <!--Определение альтернативного экрана оплаты-->
  <screen type="Sum" sum="sum-banner"/>
</scenario>
```

Задача 3. Вывод анимированных картинок в сценарии

Решение. Использование **`decor="info"`** для вывода изображения из help, **`decor="simple"`** — для рекламы из кабинета

```
...
<screen type="Digital"
        decor="info"
        title="Enter 11-digit Mobile Number"
        id="first_screen">
...
```

Задача 4. Диалог (экран) после экрана оплаты с оценкой качества/удобства интерфейса.

Решение. В настройках сервиса на вкладке «Сценарий» в поле «Предлагать следующие сервисы, после оплаты данного сервиса» указать **`id`** сервиса, который будет предложен после оплаты. данный сервис должен реализовывать логику оценки качества/удобства интерфейса. Реализовано для отдельных интерфейсов. Наличие реализации необходимо уточнять у разработчиков.

13 ОСОБЕННОСТИ СИНТАКСИСА ДЛЯ ПРИЛОЖЕНИЯ «ПРИЕМ ПЛАТЕЖЕЙ ANDROID»

Для усовершенствованного обработчика реализованы экраны **`Numeric`** (раздел 4.6.28), **`Selector`** (раздел 4.6.32), **`Confirm`** (раздел 4.6.6), **`Group`** (раздел 4.6.18), **`Info`** (раздел 4.6.23) и поля ввода `<text-field>` (раздел 5.7), `<selector-field>` (раздел 5.10.1), `<numeric-field>` (раздел 5.8). Доступы следующие элементы: `<actions>` (раздел 4.3), `<action>` (раздел 4.4), `<goto>` (раздел 6.1.3), `<if>` (раздел 6.1.2), `<online-request>` (раздел 6.6), `<dialog>` (раздел 6.5), `<modify>` (раздел 6.4.2), `<redirect>` (раздел 6.11), `<set>` (раздел 6.13).

Обрабатываются поля `<date-field>` (раздел 5.11), `<checkbox-field>` (раздел 5.9).

Если значение поля не соответствует правилам валидации (раздел 5.14.1), то отображается содержимое элемента `<help>` (раздел 5.13). Аналогично для `<checkbox-field>`, в зависимости от значения атрибута **`required`**, отображается содержимое элемента `<help>` или `<help-off>`.

В системе обновлений сценарии должны размещаться в каталоге ***`<корень CO>/android_im7/forms/advanced`***.

14 МИКРОСАЙТЫ

14.1 ОБЩИЕ СВЕДЕНИЯ

В ТПО 5 и 7 версии реализована возможность встраивания микросайтов в сценарии.

Предупреждение!
Поддерживается интерфейсами Blusphere2, Blues(v7). Возможность использования в других интерфейсах уточняйте у сотрудников технической поддержки.

Микросайт — небольшое локальное приложение, написанное на HTML и JavaScript, запускаемое внутри ТПО во встроенном браузере. При этом приложение может:

1. Реализовывать внутри себя сложную бизнес-логику.

2. Обращаться к данным за пределами ТПО к некоторому серверу.

3. Работать с прокси-сервером.

4. Генерировать события, которые перехватывает ТПО для того, чтобы выполнить некоторое действие.

Результатом работы микросайта может быть:

1. Просмотр информации, навигация по микросайту, закрытие сервиса с микросайтом по событию от микросайта (выход).

2. Просмотр информации, ввод данных, отправка этих данных на внешний сервер через прокси-сервер.

3. Просмотр информации, ввод данных, передача этих данных по событию в ТПО, дальнейшая обработка события терминалом, например, оплата.

Для создания микросайта:

1. Добавьте в кабинете агента сервис с типом модуля ввода данных «SiteProvider». Модуль ввода данных « SiteProvider» предварительно должен быть добавлен в систему в разделе «Справочники — Сервисы — Модули терминалов». Сборка ТПО так же должна поддерживать обработку модуля.

Предупреждение!
Для работы 7 версии ТПО используется библиотека `org.json-20120521.jar`, которую можно запросить у разработчиков и необходимо поместить в каталог ***<корень ТПО>/lib***.

2. Добавьте микросайт на ТПО, используя систему обновлений. В 7 версии ТПО все файлы сайта должны быть заархивированы. Архив обязательно должен быть в формате .zip. Название архива — `site.zip`. В корне архива должен располагаться файл `index.html`. Архив необходимо расположить в каталоге ***<корень ТПО>/res/module/input/site/<код сервиса>/***. Для ТПО 5 версии сайт со всеми файлами (название файла `index.html`) расположите в каталоге ***<корень ТПО>/resources/sites/<код сервиса>/***.

Web: https://soft-logic.ru/
Mail: info@soft-logic.ru

14.2 ПРОТОКОЛ ДЛЯ ВЗАИМОДЕЙСТВИЯ ЛОКАЛЬНОГО САЙТА С ТПО

Для взаимодействия локального сайта с ТПО реализован следующий протокол :

1. Функция выхода в главное меню (действие соответствующее кнопке **«Выход»**) — `Terminal.toMain()`.

2. Функция возврата на предыдущее меню (действие соответствующее кнопке **«Назад»** на первом экране) — `Terminal.toBack()`;

3. Функция редиректа (перехода) на другой сервис — `Terminal.toService(int id)`, **`id`** — идентификатор сервиса. При редиректе на другой сервис передача атрибутов, введенных на страницах микросайта, не поддерживается.

4. Функция печати — `Terminal.toPrinter(String template, String data)`, где:

   1) **`template`** — имя шаблона в ***`<корень ТПО>\res\templates\<локаль>\custom\`***;

   2) **`data`** — строка данных, представляет собой JSON объект вида '{"key1":"value1","key2":"value2", ... , "keyN":"valueN" }'. Поля данного объекта доступны в шаблоне чека как $key1, $key2, ... , $keyN.

5. Функция перехода к оплате — `Terminal.toPay(String data, int sumOperation)`, где:

   1) **`sumOperation`** — фиксированная сумма операции в копейках (передавать 0, если не используется);

   2) **`data`** — JSON массив InputElement '[{"key":"key1", "keytitle":"keytitle1", "value":"value1", valuetitle:"valuetitle1"}, ... ,{"key":"keyN", "keytitle":"keytitleN", "value":"valueN", valuetitle:"valuetitleN"}]'. У InputElement могут отсутствовать

поля **keytitle** и **valuetitle**, тогда им присваиваются значения полей **key** и **value**.

Предупреждение!
При копировании примера для тестирования не забудьте удалить лишние разрывы строк в блоке отрисовки элементов, то есть, например, <input type="button" value="Перейти на другой сервис" onclick="toService()">

Пример html-файла:

```
<HTML>
  <HEAD>
    <TITLE>Тестовая страница для обработчика локальных сайтов</TITLE>
    <meta http-equiv="Content-Type" content="text/html; charset=Utf-8">
      <script>
        function toMain() {
          Terminal.toMain();
        }
        function toBack() {
          Terminal.toBack();
        }
        function toPay(){
          Terminal.toPay('[{"key":"id1", "value":"89235678993"},
                         {"key":"id2", "value":"89000000000"}]',15000);
        }
        function toPrinter(){
          Terminal.toPrinter('test','{"str":"Строка для печати"}');
        }
        function toService(){
          Terminal.toService(3);
        }
      </script>
  </HEAD>
  <BODY>
    <H1>Тестовая страница для обработчика локальных сайтов </H1><br>
      <img src="img/logo.png">
      <br><br>
      <p><input type="button" value="Выйти в главное меню"
                onclick="toMain()"></p><br>
```

```
      <p><input type="button" value="Назад" onclick="toBack()"></p><br>
      <p><input type="button" value="Перейти на оплату"
                onclick="toPay()"></p><br>
      <p><input type="button" value="Распечатать информацию"
                onclick="toPrinter()"></p><br>
      <p><input type="button" value="Перейти на другой сервис"
                onclick="toService()"></p><br>
      <p><a href="2.html">Переход на другую страницу
                          из архива</a></p><br>
  </BODY>
</HTML>
```

Пример файла 2.html:

```
<HTML>
 <HEAD>
 <TITLE></TITLE>
 <meta http-equiv="Content-Type" content="text/html; charset=Utf-8">
 </HEAD>
 <BODY>
 <H1>Вторая страница, лежащая в том же архиве</H1><br>
 <img src="img/logo.png">
 <br><br>
 <p>Вторая страница</p><br>
 <a href="index.html">Обратно на первую</a>
 </BODY>
</HTML>
```

Предупреждение!
Обе страницы должны размещаться в корне архива.

14.3 ИСПОЛЬЗОВАНИЕ ПРОКСИ-СЕРВЕРА

Если требуется использовать прокси-сервер, то его необходимо настроить в конфигураторе ТПО (для 5 версии ТПО — рисунок 14.3.1, для 7 версии ТПО — рисунок 14.3.2).

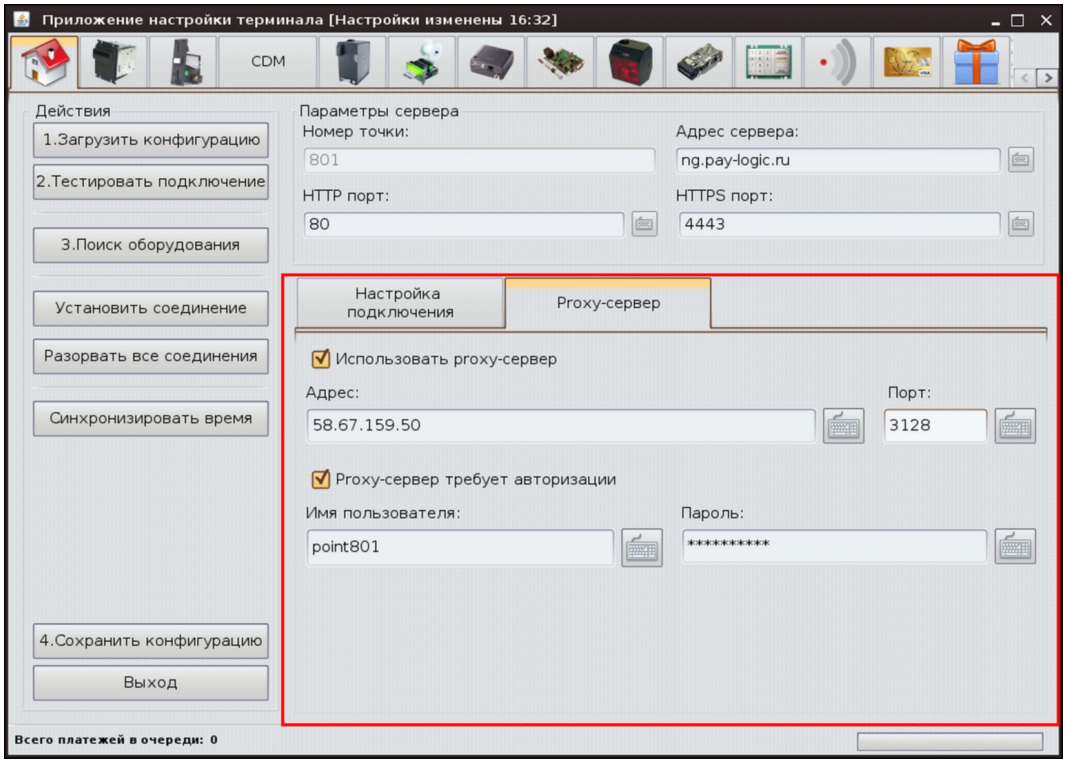


Рисунок 14.3.1 — Настройка proxy-сервера в 5 версии ТПО

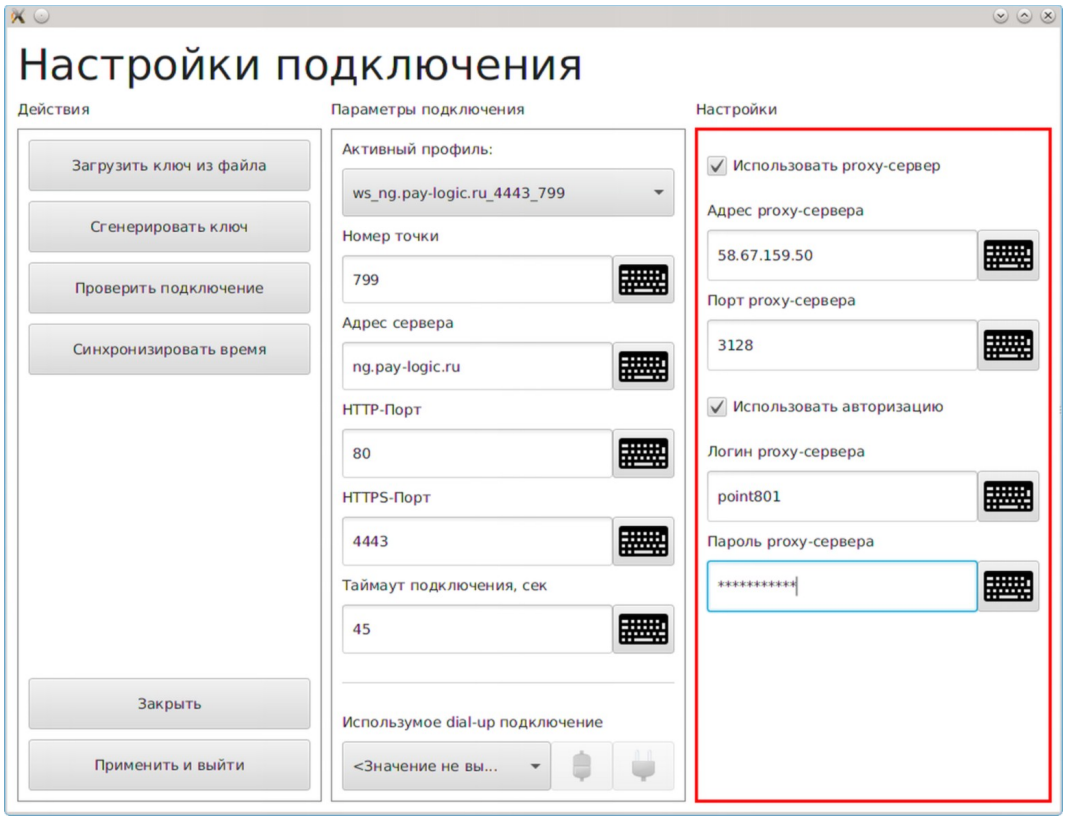


Рисунок 14.3.2 — Настройка proxy-сервера в 7 версии ТПО

14.4 ПРИМЕР НАСТРОЙКИ

Создадим сервис с соответствующим типом обработчика (рисунок 14.4.1). Идентификатор сервиса — 1290.

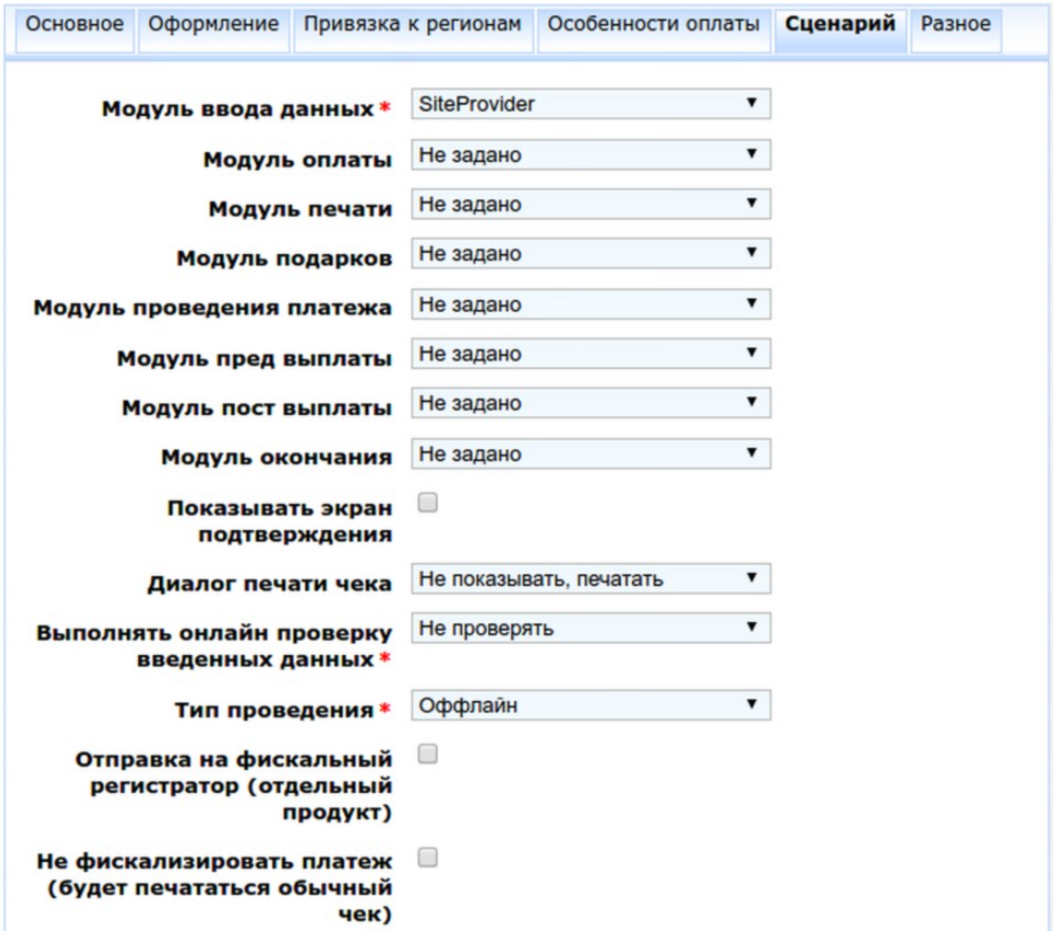


Рисунок 14.4.1 — Свойства сервиса

В 7 версии поместим в каталог **`./res/module/input/site/1290/`** архив `site.zip` с файлом `index.html`. Код `index.html` соответствует примеру из раздела 14.2.

При выборе сервиса в меню будет отображен следующий экран (рисунок 14.4.2).

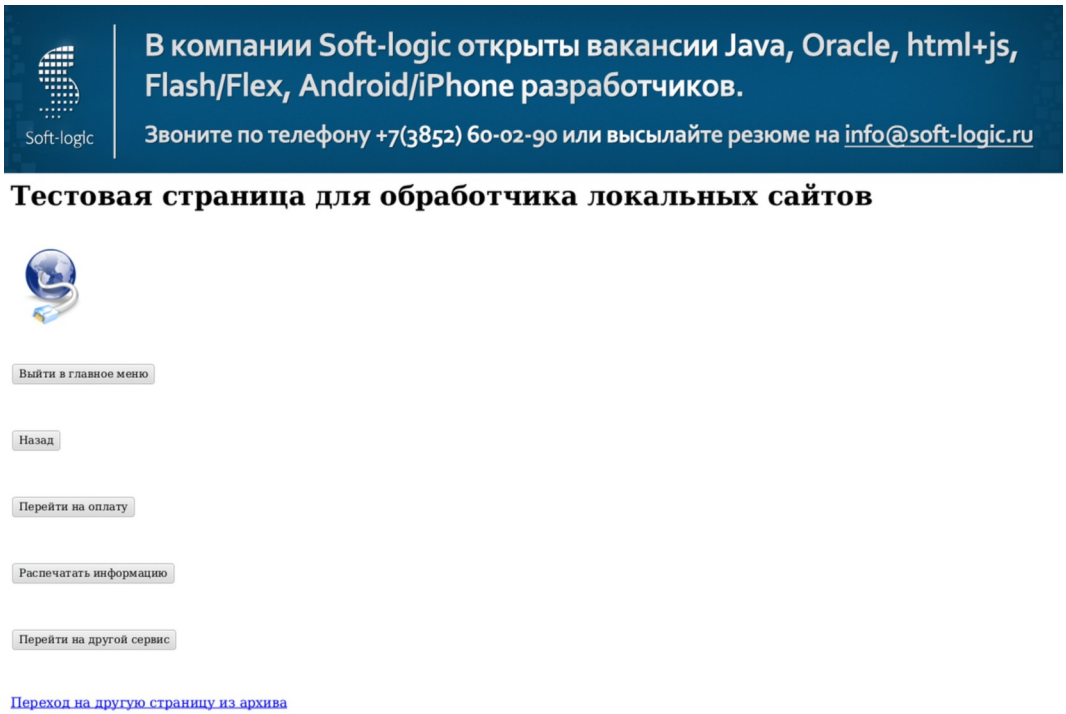


Рисунок 14.4.2 — Пример микросайта

При переходе к оплате на экран передаются заголовок и значение переменной **`id1`**, сумма к оплате «153 руб.». Согласно настройкам взимается комиссия в размере 3 руб. — рисунок 14.4.3.

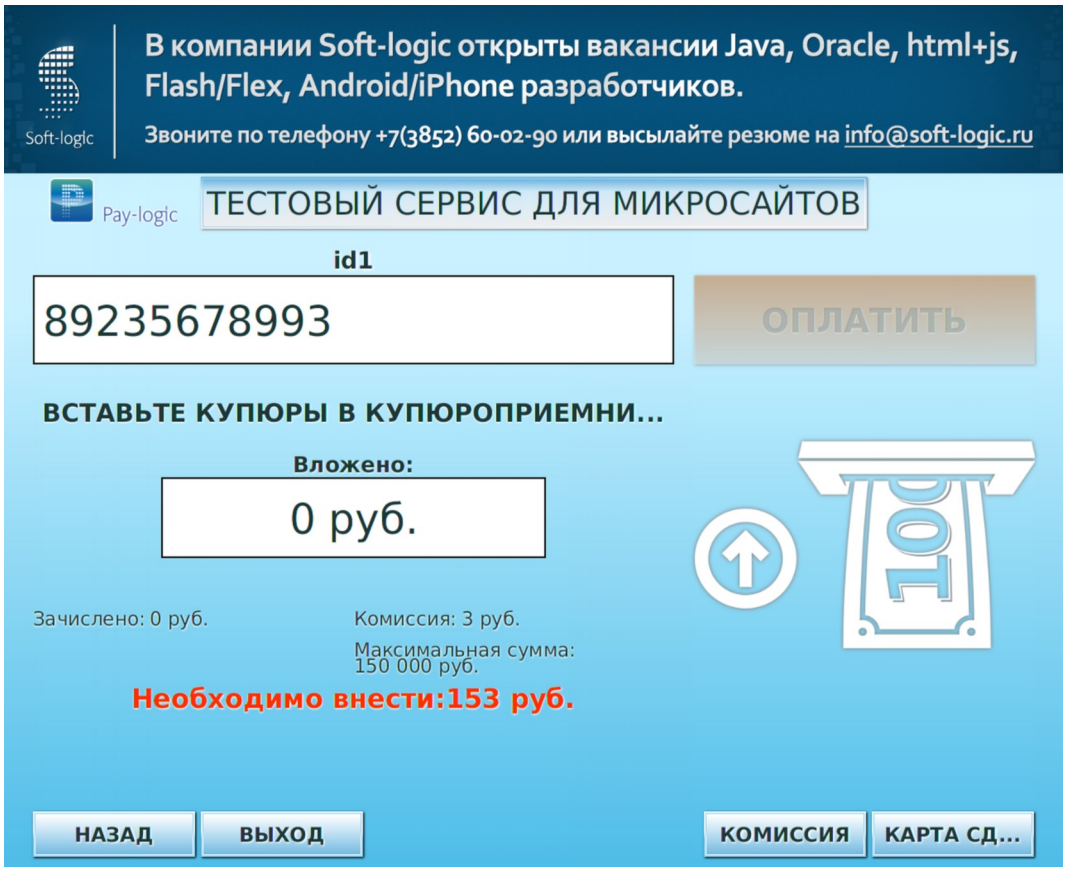


Рисунок 14.4.3 — Экран оплаты

Чтобы осуществить переход к экрану оплаты с данными, введенными клиентом, необходимо реализовать ввод данных и их передачу в функцию `toPay()` с помощью JavaScript в соответствии с указанным выше форматом.

В следующем примере микросайт предоставляет возможность перехода на внешний ресурс (рисунок 14.4.4).

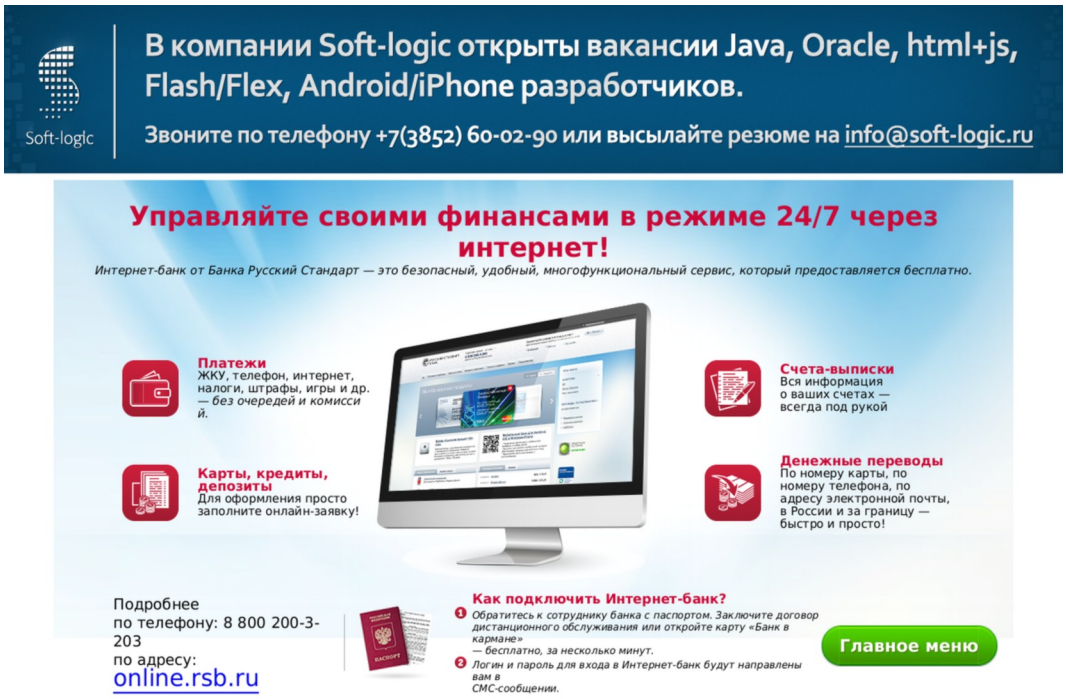


Рисунок 14.4.4 — Пример микросайта с возможностью перехода на внешний сайт

После перехода на внешний сайт возврат в главное меню терминала будет осуществлен по истечении **«Таймаута выхода в стартовую страницу при навигации внутри меню»**, установленного в параметрах точки.

В 5 версии ТПО обработка JavaScript не поддерживается.

14.5 ИСПОЛЬЗОВАНИЕ МИКРОСАЙТОВ В СИСТЕМЕ ЭЛЕКТРОННЫХ КОШЕЛЬКОВ

В связи с тем, что платформа электронных кошельков наследует сервисы из процессинга, то в процессинге добавьте сервис с типом обработчика «SiteProvider». Затем разместите на сайте страницу */site/id_сервиса/index.html* с необходимым содержимым. Например,

```
<H2>What is SmartKeeper?</H2>
<P>SmartKeeper is an online payment platform that lets you make easy
online payments for products and services using different payment ways
like online money, bankcards, and payment terminals from almost anywhere
in the world.</P>
<H2>Can I use my wallet without registering as the system user?</H2>
<P>Yes, you can make some payments for a limited number of services using
as payment source any bankcard.</P>
```

При выборе сервиса созданная страница будет открыта в новом фрейме.

Для перехода на произвольный сайт (страницу) используйте усовершенствованный тип обработчика и инструкцию `<open-url>`. Подробнее в документе «Сценарии оплаты для усовершенствованного модуля ввода данных. Программное обеспечение «Процессинговый центр Pay-logic». Руководство пользователя».