SOFT-LOGIC

Paylogic

Настройка универсального шлюза. Программное обеспечение «Процессинговый центр Pay-logic»

Руководство администратора

АННОТАЦИЯ

Описывает процесс настройки универсального шлюза программного обеспечения «Процессинговый Центр Pay-logic»

Версия руководства: 1.22

Руководство актуально для «Процессингового центра Pay-logic» версий 5.3.x

ООО «Софт-Лоджик»

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

Тел: (3852) 72-27-27

ОГЛАВЛЕНИЕ

Web: http://www.pay-logic.ru/
Mail: info@soft-logic.ru

ИСТОРИЯ ИЗМЕНЕНИЙ

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.0

Дата публикации: 29.07.2014.

Изменение	Раздел
Общие улучшения:	
Перенос документа из инструкции по установке процессинга	-

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.1

Дата публикации: 30.07.2014.

Изменение	Раздел
Общие улучшения:	
Актуализация описания добавленного функционала (использование ЭЦП, использование JavaScript, использование внешних Velocity-параметров, запрос баланса, определение HTTP-заголовков, селекторы в advanced-запросах и т.п.)	-

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.2

Дата публикации: 01.08.2014.

Изменение	Раздел
Новый функционал:	
Внедрена возможность определения кодов ответов для каждого типа запроса	3.2.5
Внедрены отдельные коды для этапа подтверждения платежа	4.3
Внедрена возможность возврата и сохранения параметров при платеже, подтверждении платежа и запросе статуса	4.2, 4.3, 4.4

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.3

Дата публикации: 26.08.2016.

Изменение	Раздел
Дополнения в документации к ранее имевшемуся функционалу:	
Добавлено описание запроса отмены платежа, запроса статуса отмены платежа	4.7, 4.8
Общие улучшения:	
Оформление документа приведено к корпоративному стандарту	-

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.4

Дата публикации: 12.09.2016.

Изменение	Раздел
Дополнения в документации к ранее имевшемуся функционалу:	
Добавлен пример указания параметра `thread-count`	3.1

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.5

Дата публикации: 07.10.2016.

Изменение	Раздел
Улучшения имеющегося функционала:	
Обновлено описание использования системы логгирования. Введены новые атрибуты системы логгирования	3.2.4

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.6

Дата публикации: 24.01.2017.

Изменение	Раздел
Улучшения в версии 4.4.2:	

Изменение	Раздел
Добавлена возможность сохранения времени обработки у поставщика	3.2.1, 4.2, 4.4
Добавлена возможность указывать номер сервиса провайдера	4.1

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.7

Дата публикации: 21.07.2017.

Изменение	Раздел
Дополнения в документации к ранее имевшемуся функционалу:	
Добавлено описание параметра **`max_conn`**, определяющего максимальное количество соединений на хост и максимальное количество соединений экземпляра HttpClient (на все хосты) одновременно, секции `<servers>`	3.2.1, 3.2.2

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.8

Дата публикации: 26.02.2018.

Изменение	Раздел
Дополнения в документации к ранее имевшемуся функционалу:	
Добавлено описание параметра **render-loader**, позволяющего использовать рендеры	4.15

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.9

Дата публикации: 31.05.2018.

Изменение	Раздел
Дополнения в документации к ранее имевшемуся функционалу:	
Исправлен пример использования хэш-генератора	4.11

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.10

Дата публикации: 09.07.2018.

Изменение	Раздел
Исправленные в документации несоответствия и ошибки:	
Исправлен пример использования пример status-timeprocess-path	4.4

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.11

Дата публикации: 27.07.2018.

Изменение	Раздел
Исправленные в документации несоответствия и ошибки:	
Исправлен пример запроса баланса	4.6

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.12

Дата публикации: 31.08.2018.

Изменение	Раздел
Улучшения в версии 4.6.9:	
Добавлен параметр для параметр для использования таймзоны даты	4.9
Добавлены параметры для использования в строковых шаблонах для таймзоны даты	4.10
Дополнения в документации к ранее имевшемуся функционалу:	
Добавлена информация по запуску шлюза	3.3

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.13

Дата публикации: 17.05.2019.

Изменение	Раздел
Улучшения в версии 4.8.1:	
Добавлен атрибут для вывода параметров товаров и услуг	4.11

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.14

Дата публикации: 11.09.2019.

Изменение	Раздел
Дополнения в документации к ранее имевшемуся функционалу:	
Добавлена возможность извлечения атрибутов из текстовых ответов на запросы проведения и проверки	4.17

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.15

Дата публикации: 30.09.2019.

Изменение	Раздел
Дополнения в документации к ранее имевшемуся функционалу:	
Добавлена возможность получения ответов на различные запросы в формате JSON	4.18

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.16

Дата публикации: 19.02.2020.

Изменение	Раздел
Дополнения в документации к ранее имевшемуся функционалу:	
Добавлено описание извлечения кодов ответов поставщика с помощью регулярных выражений	3.2.5
Добавлено описание последовательного вызова нескольких advanced-функций	4.5.1
Добавлено описание передачи параметров платежа в URL-запросе	4.19
Добавлено описание методов, атрибутов и параметров для работы с банковскими операциями	4.11.1, 4.10

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.17

Дата публикации: 10.03.2020.

Изменение	Раздел
Дополнения в документации к ранее имевшемуся функционалу:	
Добавлено описание значения по умолчанию для параметра **`pay-url`**, которое необходимо указать для успешного проведения платежа	4.2
Добавлено описание параметров для передачи авторизационных данных пользователя в заголовке HTTP-запроса	4.1
Добавлен раздел, описывающий возможность отключения офлайн-проверки реквизитов платежа на стороне сервера	4.21

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.18

Дата публикации: 17.04.2020.

Изменение	Раздел
Улучшения в версии 4.9.7:	
Добавлена возможность сохранить результат выполнения запроса проведения платежа перед его отдачей в запрос подтверждения	4.23
Дополнения в документации к ранее имевшемуся функционалу:	
Добавлена возможность получения от поставщика информации о комиссии и ее настройках	4.22

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.19

Дата публикации: 29.05.2020.

Изменение	Раздел
Исправленные несоответствия и ошибки:	
Исправлены названия параметров запроса подтверждения платежа	4.4

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.20

Дата публикации: 30.11.2020.

Изменение	Раздел
Общие улучшения в документе:	
Добавлен раздел «Экранирование спецсимволов в запросах».	4.13

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.21

Дата публикации: 26.11.2021.

Изменение	Раздел
Дополнения в документации к ранее имевшемуся функционалу:	
Добавлено описание атрибута **`$payment.providerDate`**	4.12

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.22

Дата публикации: 28.06.2022.

Изменение	Раздел
Дополнения в документации к ранее имевшемуся функционалу:	
Актуализировано описание секции указания логгера шлюза	3.2.4

1 ВВЕДЕНИЕ

Данное руководство предназначено для описания процесса настройки универсального шлюза для программного обеспечения «Процессинговый центр Pay-logic» версии 4.x.x. Универсальный шлюз написан на языке программирования Java и представляет собой модуль, входящий в состав ядра системы шлюзов.

Универсальный шлюз позволяет:

1. Настраивать проведение платежей онлайн поставщикам услуг, работающим по XML-протоколам, GET/POST-протоколам, текстовым или бинарным протоколам.
2. Настраивать проверку реквизитов, в том числе онлайн-проверки с точек приема платежей.
3. Настраивать произвольные запросы усовершенствованного обработчика, для вызова их с точек приема платежей и возврата различных параметров для анализа их на клиенте.
4. Настраивать запрос баланса у поставщика услуг, если схема работы с ним построена по авансовому механизму.

2 ПРЕДПОЛАГАЕМАЯ АУДИТОРИЯ

Предполагается, что настройка будет осуществляться человеком, знакомым со следующими областями и технологиями:

1. Знание XML для написания конфигурационного файла шлюза, а также пониманием XPath-запросов.
2. Владение знаниями по настройке и установке шлюзов процессинга.

Помимо этого, для развертывания на операционных системах, отличных от Windows (Solaris, Linux, etc.), необходимы знания и навыки работы с shell-интерпретатором и знание базовых Unix-команд.

3 НАСТРОЙКА УНИВЕРСАЛЬНОГО ШЛЮЗА

3.1 ОБЩИЕ ПОЛОЖЕНИЯ

Большинство поставщиков услуг (ПУ) можно подключить к процессингу с помощью универсального шлюза.

Под это категорию попадают поставщики, работающие по протоколу ОСМП, Киберплат или любому другому протоколу, по которому в одном запросе передается один платеж, параметры платежа передаются с помощью HTTP-параметров либо в XML-формате в теле запроса.

В данном случае подключение сводится к настройке шлюза с помощью конфигурационного файла.

Существуют две реализации универсального шлюза:

1. **Simple** — более ранняя реализация, обрабатывает запросы в один поток. Предоставляет меньше возможностей, чем Extended. В качестве класса фабрики универсального шлюза следует указать класс:

```
ru.softlogic.processing.gates.universal.Factory
```

2. **Extended** — более поздняя реализация. Может обрабатывать запросы в несколько потоков. Количество потоков определяется значением параметра **`thread-count`** (по умолчанию равен 1). Указывается в конфигурационном файле. Пример:

```
<param name="thread-count" param="10"/>
```

Предоставляет все возможности **simple** и некоторые дополнительные. В качестве класса фабрики универсального шлюза следует указать класс:

```
ru.softlogic.processing.gates.universal.v2.Factory
```

3.2 КОНФИГУРАЦИЯ УНИВЕРСАЛЬНОГО ШЛЮЗА

3.2.1 ОБЩИЕ СВЕДЕНИЯ

Конфигурационный файл шлюза состоит из следующих разделов:

1. Секция настройки серверов поставщика услуги.
2. Секция задания класса API-шлюза.
3. Секция указания логгера.
4. Секция указания кодов ответов.
5. Секция задания параметров шлюза.

Пример конфигурационного файла:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Протокол ОСМП -->
<config>
 <servers max_conn="50">
  <!-- max_conn - определяет максимальное количество соединений на хост и
                  максимальное количество соединений экземпляра
                  HttpClient (на все хосты) одновременно-->
  <!-- Секция с указанием адресов ПУ, их может быть несколько и при
       недоступности какого-либо, платежи будут проводиться через другие.
       Для сервера задаются:
        1) host - адрес
        2) port - порт
        3) scheme - http или https
       Внутри сервера могут быть следующие элементы:
        1) auth-basic - если используется Basic авторизация, с
           атрибутами username и password, например <auth-basic
           username="paylogic" password="123"/>
        2) keystore - если используется клиентский сертификат, с
           атрибутами path (путь), password (пароль) и type (тип: PKCS12
           или JKS), например <keystore path="/home/cert.p12"
           password="123" type="PKCS12"/>
```

```
 -->
 <server host="server.ru" port="443" scheme="https"> </server>
</servers>

<!-- Класс API, указать в зависимости от используемой реализации-->
<api class="ru.softlogic.processing.gates.universal.v2.ExtendedUniversalApi"/>

<!-- Название логгера для логов -->
<logger name="uni"/>

<!-- Коды ответов по действиям, из каких кодов ставить в успех, из
     каких в ошибку, из каких в оборабоку и неизвестный статус -->
<codes success="0" error="1,2,3,4,5,6" process="" unknown="-10000"/>

<gate-config>
 <params>

  <!-- Url и параметры для check, возможные значения параметров для
       всех методов одинаковые и могут быть:
       id1, id2,... - данные из формы
       id - номер транзакции
       date - дата и время в формате, задается ниже
       realSum - сумма в рублях в формате 10.00
       sum - сумма в копейках
       point - id точки
       check - номер чека
  -->
  <param name="check-url" param="/index.php"/>
  <param name="check-params" param="action=check;number=#id1#"/>

  <!-- Путь к результату ответа в xml -->
  <param name="check-result-path" param="response/code"/>

  <!-- Url и параметры платежа -->
  <param name="pay-url" param="/index.php"/>
  <param name="pay-params" param="action=payment;receipt=#id#;
                          date=#date#;number=#id1#;amount=#realSum#"/>

  <!-- Путь к результату ответа в xml -->
  <param name="pay-result-path" param="response/code"/>
```

```
   <!-- Путь к транзакции, задается опционально, если провайдер ее
        возвращает -->
   <param name="pay-transaction-path" param="response/authcode"/>

   <!-- При необходимости проверка статуса, если не нужна, то не
        задавать параметр status-url -->
   <param name="status-url" param="/index.php"/>
   <param name="status-params" param="action=status;receipt=#id#"/>
   <param name="status-result-path" param="response/code"/>
   <param name="status-transaction-path" param="response/authcode"/>

   <!--Запрос отмены платежа-->
   <param name="cancel-request-url" param="/url/to/cansel/"/>
   <param name="cancel-request-params"
          param="reqoestId=#reject-request-id#;summ=#reject-sum#"/>
   <param name="cancel-request-result" param="response/cancel"/>
   <param name="cancel-request-xml" param="request/cancel"/>

   <!--Запрос статуса отмены платежа-->
    <param name="cancel-status-url" param="/url/to/cansel/"/>
    <param name="cancel-status-params"
           param="reqoestId=#reject-request-id#;summ=#reject-sum#"/>
    <param name="cancel-status-result" param="response/status"/>
    <param name="cancel-status-xml" param="request/status"/>

   <!-- Таймаут обработки шлюза -->
   <param name="timeout" param="20"/>

   <!-- Формат даты и времени -->
   <param name="date-format" param="yyyy-MM-dd'T'HH:mm:ss"/>

<!-- Формат даты и времени обработки операции у поставщика -->
<param name="response-date-format" param="yyyy-MM-dd'T'HH:mm:ss"/>

   <!-- Метод отправки GET или POST -->
   <param name="method" param="GET"/>
   <param name="encode" param="false"/>
   <param name="response-charset" param="cp1251"/>

  </params>
 </gate-config>
</config>
```

3.2.2 СЕКЦИЯ НАСТРОЙКИ СЕРВЕРОВ ПОСТАВЩИКА УСЛУГ

При необходимости в секции `<servers>` возможно задать максимальное количество соединений на хост и максимальное количество соединений экземпляра HttpClient (на все хосты) одновременно, указав параметр **`max_conn`**.

В секции `<server>` в простейшем случае задается только IP-адрес сервера ПУ, порт и схема обращения к ресурсу: https или http.

Пример:

```
<servers max_conn="50">
 <server host="server.ru" port="443" scheme="https">
 </server>
</servers>
```

В случае дополнительной авторизации на сервере ПУ, можно использовать вложенные элементы, такие как:

1. `auth-basic` для Basic-авторизации.
2. `keystore` для авторизации с помощью клиентских сертификатов.

Пример:

```
<servers>
 <server host="server.ru" port="443" scheme="https">
   <auth-basic username="paylogic" password="123"/>
   <keystore path="/home/cert.p12" password="123" type="PKCS12"/>
 </server>
</servers>
```

Тип хранилища сертификатов может быть двух видов: **PKCS12** либо **JKS** (Java KeyStore Format).

В случае использования Proxy-сервера, его реквизиты указываются в элементе `server` в атрибутах:

1. **`proxyHost`** — адрес сервера.

2. **proxyPort** — порт сервера.
3. **proxyUser** — пользователь для авторизации на Proxy сервере.
4. **proxyPassword** — пароль для авторизации.

3.2.3 СЕКЦИЯ ЗАДАНИЯ КЛАССА API ШЛЮЗА

Для реализации шлюза **Simple** в качестве класса API шлюза должен быть указан:

```
ru.softlogic.processing.gates.universal.SimpleUniversalApi
```

Задается в элементе api:

```
<api class="ru.softlogic.processing.gates.universal.SimpleUniversalApi"/>
```

Для реализации шлюза **Extended** в качестве класса API шлюза должен быть указан:

```
ru.softlogic.processing.gates.universal.v2.ExtendedUniversalApi
```

Задается в элементе api:

```
<api
class="ru.softlogic.processing.gates.universal.v2.ExtendedUniversalApi"/>
```

3.2.4 СЕКЦИЯ УКАЗАНИЯ ЛОГГЕРА ШЛЮЗА

Название логгера задается в элементе <logger>.

```
<logger name="uni" level="INFO" secured="false"/>
```

В случае отсутствия элемента логгера будет использоваться логгер с именем **universal**.

Предварительно можно настроить сам логгер в файле *log4j.properties* согласно документу «Обслуживание программного обеспечения «Процессинговый центр Pay-logic». Руководство администратора».

Настраивать логгер в *log4j.properties* необязательно, в случае отсутствия там секции логгирование будет происходить автоматически по пути:

```
<gates home>/log/gate/<logger name>/<log files>
```

Параметр **level** определяет уровень логгирования. В случае отсутствия параметра уровень будет DEBUG.

Параметр **secured** определяет, будут ли отображаться логи платежа в кабинете. По умолчанию он равен **true** — это означает, что логи в бэк-офисе просмотреть нельзя, они защищены. При изменении значения на **false** пользователь с правами администратора сможет увидеть логи в разделе кабинета «Диспетчерская — Поиск платежа» в окне подробной информации о платеже (рисунок 1).

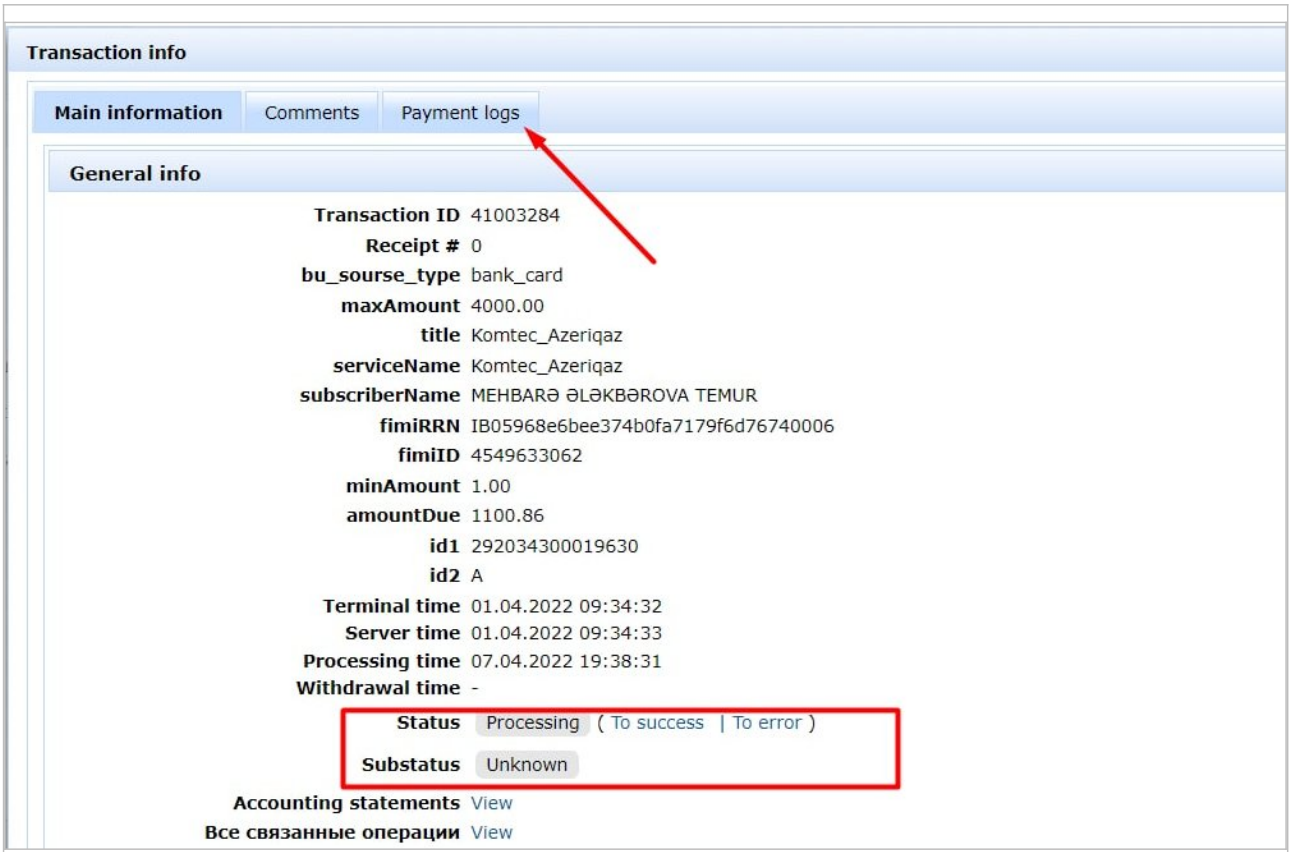

Рисунок 1 — Отображение логов в разделе подробной информации о платеже

Если в логах присутствуют пароли в открытом виде, номера банковских карт и т.д., то содержимое следует защитить. Для этого установите значение **secured=false** и задайте правило очистки логов от закрытых данных с помощью атрибутов **clear-regexp** и **clear-replace**, которые задают регулярное выражение замены и на что менять найденые вхождения.

Предположим, запрос выглядит так:

```
?username=test&password=98439dj34&account=1111111
```

Тогда правила замены задаем так:

```
<logger name="uni" level="INFO" secured="false" clear-
regexp="password=[^&amp;]+" clear-replace="password=password"/>
```

В случае отсутствия правил замены и указании **secured=false** все данные в логах будут отображаться в бэк-офисе в открытом виде.

3.2.5 СЕКЦИЯ УКАЗАНИЯ КОДОВ ОТВЕТОВ

Для обновления статуса платежа необходимо задать реакцию на определенные возвратов ПУ.

Это делается с помощью элемента <codes>:

```
<codes success="0" error="1,2,3,4,5,6,-10001" process=""
       unknown="-10000"/>
```

Используются следующие атрибуты:

1. В атрибуте **success** задаются коды через запятую, при которых платеж будет ставиться в успех.
2. В атрибуте **error** задаются коды через запятую, при которых платеж будет ставиться в статус «Проведение — Ошибка».
3. В атрибуте **process** задаются коды через запятую, при которых платеж будет ставиться в статус «Проведение — Проводится».
4. В атрибуте **unknown** задаются коды через запятую, при которых платеж будет ставиться в статус «Проведение — Неизвестен».
5. В атрибуте **confirm** задаются коды через запятую, при которых платеж будет ставиться в статус, требующий подтверждения.

Все атрибуты опциональны. В случае сетевых ошибок при проведении шлюз возвращает код **-10000**, этот код следует указать в атрибуте **unknown**. В случае сетевых ошибок при проверке реквизитов платежа, шлюз возвращает **-10001**, этот код следует указать в атрибуте **error**.

В случае, если ПУ возвращает код, отличный от заданных в атрибутах, платеж ставится в статус «Проведение — Неизвестен».

Если ПУ возвращает не числовые коды ответов, а, например, текстовые, то необходимо задать маппинг текстовых кодов в числовые, с помощью атрибута **mapping**.

Пример:

```
<codes success="0" error="10" process="1" unknown="-10000"
       mapping="OK=0,PROCESS=1,ERROR=10"/>
```

При необходимости, коды ответов можно задавать для каждого типа запросов, указывая соответствующий префикс, например:

```
<check-codes success="0" error="10" process="1" unknown="-10000"
             mapping="OK=0,PROCESS=1,ERROR=10"/>
<pay-codes success="1" error="11" process="1" unknown="-10000"
           mapping="OK=0,PROCESS=1,ERROR=10"/>
```

Внимание!
Реализация **Simple** не поддерживает возможность указания кодов ответов для типов запросов.

В случае определения кодов для типа запроса, именно они будут анализироваться на этапе выполнения конкретного запроса, коды по умолчанию для этого типа запроса роли играть не будут.

С версии 4.9.0 добавлен атрибут **regex-mapping**, который позволяет извлекать коды ответов поставщика с помощью регулярных выражений. При этом регулярные выражения указываются через символ «;».

Рассмотрим пример. Пусть в случае успеха шлюз возвращает любое число, не равное 0, а в случае ошибки — строку, содержащую текст ошибки либо 0. Задание кодов ответов для данного примера приведено ниже.

Пример:

```
<pay-codes success="1" error="-10001,0" unknown="-10000"
           regex-mapping="[a-zA-Z]+=0;[1-9]|\d{2,}=1"/>
```

3.2.6 СЕКЦИЯ УКАЗАНИЯ ПАРАМЕТРОВ ШЛЮЗА

Параметры шлюза можно разделить на 7 категорий:

1. Параметры запроса проверки номера абонента.
2. Параметры запроса проведения платежа.
3. Параметры запроса подтверждения платежа (опционально).
4. Параметры запроса статуса платежа.
5. Параметры запросов усовершенствованного обработчика.
6. Параметры запроса баланса.
7. Общие параметры.

3.3 ЗАПУСК ШЛЮЗА

Для запуска универсального шлюза конфигурационный файл нужно разместить на сервере в каталоге:

home/gates/configs/<каталог_с названием_созвучным_названию_шлюза>/<имя_конфига.xml>.

Затем в главном конфигурационном файле шлюзов необходимо добавить новую запись в разделе *<gates>*, указав: путь до шлюза, ID провайдера, фабрику шлюза. Главный конфигурационный файл шлюзов находится в каталоге **home/gates/config.xml**.

```
<gates>
      <gate enable="true"
            factory="ru.softlogic.processing.gates.universal.Factory"
            id-provider="1690"
            config="/home/gates/configs/gate/gate.xml"/>
</gates>
```

В разделе *<gates>* основного конфигурационного файла шлюзов каждому шлюзу соответствует элемент *<gate>*. В элементе <gate> указываются следующие атрибуты:

1. **enable** — флаг активности шлюза.
2. **factory** — названия класса фабрики шлюза.
3. **id-provider** — ID поставщика услуг в системе (раздел в кабинете процессинга «Провайдеры — Провайдеры»).
4. **config** — путь к конфигурационному файлу шлюза.

Затем необходимо настроить журналирование работы шлюза. Для этого в файл *log4j.properties* необходимо добавить секцию следующего содержания:

```
log4j.logger.название_шлюза=debug, название_шлюза
log4j.appender.название_шлюза =org.apache.log4j.DailyRollingFileAppender
log4j.appender.название_шлюза.DatePattern='.'yyyy-MM-dd
log4j.appender.название_шлюза.File=${app.dir}/log/gate/beeline/beeline .l
og
log4j.appender.название_шлюза.layout=org.apache.log4j.PatternLayout
log4j.appender.название_шлюза.layout.conversionPattern=%d{ABSOLUTE} %5p
%t %c{1}:%M:%L - %m%n
```

После выполнения всех процедур нужно перезапустить систему шлюзов, выполнив команду: `service paylogic restart`.

Предупреждение!
При выполнении команды будут остановлены и запущены все шлюзы.

4 ПАРАМЕТРЫ ШЛЮЗА

4.1 ПЕРЕДАЧА АВТОРИЗАЦИОННЫХ ДАННЫХ В ЗАГОЛОВКЕ HTTP-ЗАПРОСА

Для того, чтобы выполнить авторизацию на сервере, пользователю необходимо отправить запрос на авторизацию и указать в нем логин и пароль, которые были выданы ему провайдером. Данные для авторизации задаются в поле **Authorization** HTTP-заголовка запроса.

Пример:

```
Authorization: Basic base64(login:password)
```

где **login** и **password** — значения логина и пароля пользователя.

Для того, чтобы передавать в заголовке HTTP-запроса авторизационные данные, которые рассчитываются автоматически, укажите следующие параметры:

```
<param name="basic-no-challenge.user" param="login"/>
<param name="basic-no-challenge.password" param="password"/>
```

где **login** и **password** — логин и пароль пользователя, выданные провайдером.

4.2 ЗАПРОС ПРОВЕРКИ НОМЕРА АБОНЕНТА

Параметры запроса проверки номера абонента в большинстве случаев представляют следующий блок:

```
<param name="check-url" param="/index.php"/>
<param name="check-params" param="action=check;number=#id1#"/>

<!-- Путь к результату ответа в xml -->
<param name="check-result-path" param="response/rcode"/>
```

Таким образом, в качестве параметров запроса проверки номера абонента указываются:

1. **check-url** — относительный URL-запроса.
2. **check-params** — GET или POST-параметры запроса через точку с запятой «;».
3. **check-result-path** — путь к результату в XML-ответе ПУ (например, ответ **<response><rcode>0</rcode></response>**).

В случае отсутствия параметра **check-url**, подразумевается, что ПУ не поддерживает проверку номера абонента.

Если ПУ возвращает в ответе не XML, а простой текст, тогда укажите в конфигурационном файле параметр **check-response-text**, в котором с помощью **#result#** определяется место ответа. Например, ответ вида **"<код>:<сообщение>"**:

```
<param name="check-url" param="/index.php"/>
<param name="check-params" param="action=check;number=#id1#"/>
<param name="check-response-text" param="#result#:#transaction#"/>
```

Дополнительно в общих параметрах укажите **text** в качестве значения параметра **response-type**.

Если ПУ возвращает в ответе параметры, которые необходимо передать клиенту, то нужно определить секцию с описанием параметров и указать расположение их значений (с помощью XPath-запросов).

Ниже приведен пример возврата двух параметров **fio** и **dolg**:

```
<param name="check-result-param.fio" param="response/fio"/>
<param name="check-result-param-title.fio" param="ФИО"/>
<param name="check-result-param.dolg" param="response/dolg"/>
<param name="check-result-param-title.dolg" param="Задолженность"/>
```

В данном случае будут возвращены два параметра со значениями из ответа ПУ, а также установленными **title**.

Возможно указывать номер сервиса провайдера в формате:

```
<param name="check-result-param.<номер сервиса провайдера>.message"
       param="//Comment"/>
<param name="check-result-param-title.<номер сервиса провайдера>.message"
       param="Информация"/>
```

В случае необходимости установки одного из параметров как суммы к оплате, в качестве его кода укажите **sum-purchase**.

Дополнительно, при проведении платежа по шлюзу полученные параметры на этапе проверки от ПУ будут доступны в контексте платежа по назначенным кодам с префиксом «**check.**», то есть, например, «**check.fio**».

4.3 ЗАПРОС ПРОВЕДЕНИЯ ПЛАТЕЖА

Запрос проведения платежа является обязательным и выглядит так:

```
<param name="pay-url" param="/index.php"/>
<param name="pay-url" param=""/>
<param name="pay-params" param="action=payment;receipt=#id#;date=#date#;
       number=#id1#;amount=#realSum#"/>

<!-- Путь к результату ответа в xml -->
<param name="pay-result-path" param="response/rcode"/>

<!-- Путь к транзакции, задается опционально, если провайдер ее
возвращает -->
<param name="pay-transaction-path" param="response/authcode"/>

<!-- Путь к дате и времени обработки запроса проведения платежа
поставщиком в XML-ответе ПУ. Задается опционально только для реализации
Extended -->
<param name="pay-transaction-path" param="response/authcode"/>
```

В качестве параметров запроса проведения платежа указываются:

1. **pay-url** — относительный URL запроса.

Внимание!

Чтобы запрос проведения платежа был успешно выполнен, укажите для параметра **pay-url** значение по умолчанию, добавив в запрос проведения следующую строку:

```
<param name="pay-url" param=""/>
```

В противном случае платеж не будет проведен.

---

2. **pay-params** — GET или POST параметры запроса через точку с запятой «;».
3. **pay-result-path** — путь к результату в XML-ответе ПУ (например, ответ **<response><rcode>0</rcode></response>**).

4. **pay-transaction-path** — путь к номеру транзакции в XML-ответе ПУ. (например, ответ **<response><authcode>12313</authcode></response>**).

5. **pay-timeprocess-path** — путь к дате и времени обработки запроса поставщиком в XML-ответе ПУ (например, **param="response/timeprocess"/>**). Параметр обрабатывается только реализацией **Extended** универсального шлюза.

Если ПУ возвращает в ответе не XML, а простой текст, тогда укажите в конфигурационном файле параметр **pay-response-text**, в котором с помощью #**result**# задается место ответа, а с помощью #**transaction**# — место транзакции.

Например, ответ вида "**<код>:<транзакция>**":

```
<param name="pay-url" param="/index.php"/>
<param name="pay-url" param=""/>
<param name="pay-params" param="action=check;number=#id1#"/>
<param name="pay-response-text" param="#result#:#transaction#"/>
```

Дополнительно в общих параметрах задайте **text** в качестве значения параметра **response-type**.

Если ПУ возвращает в ответе параметры, которые необходимо передать на клиента или сохранить в платеже, нужно определить секцию с описанием параметров и локацией их значений (с помощью XPath-запросов).

Ниже приведен пример возврата двух параметров **fio** и **dolg**.

```
<param name="pay-result-param.fio" param="response/fio"/>
<param name="pay-result-param.dolg" param="response/dolg"/>
```

В данном случае на клиента вернутся два параметра со значениями из ответа ПУ.

В случае необходимости установки одного из параметров как суммы к оплате, в качестве его кода укажите **sum-purchase**.

4.4 ЗАПРОС ПОДТВЕРЖДЕНИЯ ПЛАТЕЖА

В некоторых случаях ПУ для проведения платежа использует два запроса: создание и подтверждение. В данном случае в разделе `pay` конфигурируется создание платежа, а в разделе `confirm` конфигурируется потдверждение платежа. В случае отсутствия параметра **`confirm-url`**, запрос подтверждения выполняться не будет. Необходимым условием для вызова подтверждения платежа является также успешное выполнение создания платежа: код результата должен быть определен в секции `confirm` для типов кодов `pay` запросов или для кодов ответов по умолчанию.

Запрос подтверждения платежа выполняется сразу после запроса проведения платежа, без каких-либо задержек.

```
<param name="confirm-url" param="/index.php"/>
<param name="confirm-params" param="action=confirm;receipt=#id#;
       date=#date#;number=#id1#;amount=#realSum#"/>

<!-- Путь к результату ответа в xml -->
<param name="confirm-result-path" param="response/rcode"/>

<!-- Путь к транзакции, задается опционально, если провайдер ее
возвращает -->
<param name="confirm-transaction-path" param="response/authcode"/>
```

В качестве параметров запроса подтверждения платежа указываются:

1. **`confirm-url`** — относительный URL запроса.
2. **`confirm-params`** — GET или POST параметры запроса через точку с запятой «;».
3. **`confirm-result-path`** — путь к результату в XML-ответе ПУ (например, ответ **`<response><rcode>0</rcode></response>`**) .
4. **`confirm-transaction-path`** — путь к номеру транзакции в XML-ответе ПУ (например, ответ **`<response><authcode>12313</authcode></response>`**).

Если ПУ возвращает в ответе не XML, а простой текст, тогда укажите в конфигурационном файле параметр **confirm-response-text**, в котором с помощью #**result**# задается место ответа, а с помощью #**transaction**# — место транзакции.

Например, ответ вида "<код>:<транзакция>"

```
<param name="confirm-url" param="/index.php"/>
<param name="confirm-params" param="action=check;number=#id1#"/>
<param name="confirm-response-text" param="#result#:#transaction#"/>
```

Дополнительно в общих параметрах задайте значение **text** для параметра **response-type**.

Если ПУ возвращает в ответе параметры, которые необходимо передать на клиента или сохранить в платеже, нужно определить секцию с описанием параметров и локацией их значений (с помощью XPath-запросов).

Ниже приведен пример возврата двух параметров: **fio** и **dolg**.

```
<param name="confirm-result-param.fio" param="response/fio"/>
<param name="confirm-result-param.dolg" param="response/dolg"/>
```

В данном случае на клиента вернутся два параметра со значениями из ответа ПУ.

В случае необходимости установки одного из параметров как суммы к оплате, в качестве его кода укажите **sum-purchase**.

4.5 ЗАПРОС СТАТУСА ПЛАТЕЖА

Данный запрос не является обязательным и выполняется, если платеж находится в состоянии «Проведение — Неизвестен» (через 5 минут после последней обработки платежа) или «Проведение — Проводится» (через 30 секунд после последней обработки платежа).

В случае отсутствия **status-url**, для таких платежей будет выполняться повторно проведение платежа.

```
<param name="status-url" param="/index.php"/>
<param name="status-params" param="action=status;receipt=#id#"/>

<!-- Путь к результату ответа в xml -->
<param name="status-result-path" param="response/rcode"/>

<!-- Путь к транзакции, задается опционально, если провайдер ее
возвращает -->
<param name="status-transaction-path" param="response/authcode"/>

<!-- Путь к дате и времени обработки запроса статуса платежа поставщиком
в XML-ответе ПУ. Задается опционально только для реализации Extended -->
<param name="status-timeprocess-path" param="response/timeprocess"/>
```

В качестве параметров запроса статуса платежа указываются:

1. **status-url** — относительный URL запроса.
2. **status-params** — GET или POST параметры запроса через точку с запятой «;».
3. **status-result-path** — путь к результату в XML-ответе ПУ (например, ответ **<response><rcode>0</rcode></response>**).
4. **status-transaction-path** — путь к номеру транзакции в XML-ответе ПУ. (например, ответ **<response><trans>123123</trans></response>**).

5. **status-timeprocess-path** — путь к дате и времени обработки запроса статуса платежа поставщиком в XML-ответе ПУ (например, **param="response/timeprocess"/>**). Параметр обрабатывается только реализацией **Extended** универсального шлюза.

Если ПУ возвращает в ответе не XML, а простой текст, тогда необходимо прописать параметр **status-response-text**, в котором с помощью #**result**# задать место ответа, а с помощью #**transaction**# — место транзакции.

Например, ответ вида "**<код>:<транзакция>**":

```
<param name="status-url" param="/index.php"/>
<param name="status-params" param="action=check;number=#id1#"/>
<param name="status-response-text" param="#result#:#transaction#"/>
```

Если ПУ возвращает в ответе параметры, которые необходимо передать на клиента или сохранить в платеже, нужно определить секцию с описанием параметров и локацией их значений (с помощью XPath-запросов).

Ниже приведен пример возврата двух параметров **fio** и **dolg**.

```
<param name="status-result-param.fio" param="response/fio"/>
<param name="status-result-param.dolg" param="response/dolg"/>
```

В данном случае на клиента вернутся два параметра, со значениями из ответа ПУ.

В случае необходимости установки одного из параметров как суммы к оплате, в качестве его кода укажите **sum-purchase**.

4.6 ЗАПРОСЫ УСОВЕРШЕНСТВОВАННОГО ОБРАБОТЧИКА

В случае необходимости обработки запросов с усовершенствованного обработчика (сценарии), для каждого типа запроса (каждой функции) необходимо определить параметры отправки, а также параметры ответа, которые будут переданы обратно на клиентское устройство.

Ниже приведен пример заполнения секции для функции `test`.

```
<param name="advanced-url.test" param="/index.php"/>
<param name="advanced-params.test" param="action=check;account=#id1#"/>

<!-- Путь к результату ответа в xml -->
<param name="advanced-result-path.test" param="response/rcode"/>
```

В качестве параметров запросов универсального обработчика указываются:

1. **`advanced-url.<func>`** — относительный URL запроса для функции `func`.
2. **`advanced-params.<func>`** — GET или POST параметры запроса через точку с запятой «;» для функции `func`.
3. **`advanced-result-path.<func>`** – путь к результату в XML ответе ПУ (например, ответ **`<response><rcode>0</rcode></response>`**).

Если ПУ возвращает в ответе параметры, которые необходимо передать на клиента, нужно определить секцию с описанием параметров и локацией их значений. Ниже приведен пример возврата двух параметров **`fio`** и **`dolg`** для функции `test`.

```
<param name="advanced-result-param.test.fio" param="response/fio"/>
<param name="advanced-result-param-title.test.fio" param="ФИО"/>
<param name="advanced-result-param.test.dolg" param="response/dolg"/>
<param name="advanced-result-param-title.test.dolg"
       param="Задолженность"/>
```

В данном случае на клиента вернутся два параметра со значениями из ответа ПУ, а также установленными **`title`**.

В случае необходимости установки одного из параметров как суммы к оплате в сценарии, в качестве его кода укажите **`sum-purchase`**.

Если ПУ в ответе возвращает переменное число параметров, которые необходимо передать на клиента, например, список счетчиков в с текущими значениями:

```
<RESULT>
 <ERROR>0</ERROR>
 <SCHS>
  <SCHID>12345679</SCHID>
  <SCHP>120</SCHP>
  <SCHINFO>Холодная вода</SCHINFO>
 </SCHS>
 <SCHS>
  <SCHID>12345680</SCHID>
  <SCHP>82</SCHP>
  <SCHINFO>Холодная вода</SCHINFO>
 </SCHS>
</RESULT>
```

Необходимо определить коллекцию возвращаемых значений следующим образом (для функции `test`):

```
<param name="advanced-result-collection.test.s"
       param="RESULT/SCHS/SCHID"/>
<param name="advanced-result-collection-title.test.s"
       param="Номер счетчика"/>
<param name="advanced-result-collection-expression.test.s"
       param="text()"/>
<param name="advanced-result-collection.test.p"
       param="RESULT/SCHS/SCHP"/>
<param name="advanced-result-collection-title.test.p"
       param="Текущие показания"/>
<param name="advanced-result-collection-expression.test.p"
       param="text()"/>
<param name="advanced-result-collection.test.stype"
       param="RESULT/SCHS/SCHINFO"/>
<param name="advanced-result-collection-title.test.stype"
       param="Тип счетчика"/>
<param name="advanced-result-collection-expression.test.stype"
       param="text()"/>
```

К каждому названию параметра из коллекции добавляется индекс по порядку, и на клиента будут переданы параметры: **s1**, **p1**, **stype1**, **s2**, **p2**, **stype2**.

В качестве expression можно использовать любое XPath-выражение (в данном примере **text()** — это содержимое элемента), например, для поиска атрибута **value: "@value"**.

В случае необходимости использования экрана типа **«Selector»** на клиенте, требуется правильным образом настроить возврат списка значения для данного типа элемента.

Внимание!
Реализация **Simple** не поддерживает возможность использования экрана типа **«Selector»** на клиенте.

Как правило, **«Selector»** предполагает возврат от ПУ списка элементов выбора с конкретным набором атрибутов, например:

```
<Products>
    <Product>
        <ServiceId>1</ServiceId>
        <ProductId>3</ProductId>
        <Address>Малахова 14</Address>
        <Account>123123</Account>
    </Product>
    <Product>
        <ServiceId>2</ServiceId>
        <ProductId>3</ProductId>
        <Address>Балтийская 2</Address>
        <Account>123124</Account>
    </Product>
</Products>
```

В данном случае необходимо определить секцию разбора следующим образом (для функции `test`):

```
<param name="advanced-result-selector.test.s" param="Products/Product"/>
<param name="advanced-result-selector-param.test.s.serviceId"
       param= "ServiceId/text()"/>
<param name="advanced-result-selector-param-title.test.s.serviceId"
       param="Номер сервиса"/>
```

```
<param name="advanced-result-selector-param.test.s.address"
       param= "Address/text()"/>
<param name="advanced-result-selector-param-title.test.s.address"
       param="Адрес"/>
<param name="advanced-result-selector-param.test.s.account"
       param= "Account/text()"/>
<param name="advanced-result-selector-param-title.test.s.account"
       param="Номер счета"/>
<param name="advanced-result-selector-param.test.s.title"
       param= "Address/text()"/>
```

Тогда на клиента будет возвращен **«Selector»** с кодом **`s`**, включающий два элемента выбора с установленными надписями в атрибуте **`title`** (адрес), а также с набором из трех параметров **`serviceId`**, **`address`** и **`account`**. Наличие атрибута **`title`** обязательно к определению, как показано на примере выше.

Иногда возникает необходимость определения для селектора переменного числа параметров, например, счетчики:

```
<Products>
    <Product>
        <ServiceId>1</ServiceId>
        <ProductId>3</ProductId>
        <Address>Малахова 14</Address>
        <Account>123123</Account>
        <Meters>
            <Meter>
                <Id>123</Id>
                <Name>Хол. вода</Name>
                <Tariff>1.0</Tariff>
                <OldValue>123123</OldValue>
            </Meter>
            <Meter>
                <Id>321</Id>
                <Name>Гор. вода</Name>
                <Tariff>2.0</Tariff>
                <OldValue>123123</OldValue>
            </Meter>
        </Meters>
    </Product>
    <Product>
```

```
        <ServiceId>2</ServiceId>
        <ProductId>3</ProductId>
        <Address>Балтийская 2</Address>
        <Account>123124</Account>
        <Meters>
            <Meter>
                <Id>124</Id>
                <Name>Хол. вода</Name>
                <Tariff>1.0</Tariff>
                <OldValue>22222</OldValue>
            </Meter>
            <Meter>
                <Id>324</Id>
                <Name>Гор. вода</Name>
                <Tariff>2.0</Tariff>
                <OldValue>22222</OldValue>
            </Meter>
        </Meters>
    </Product>
</Products>
```

В данном случае возможно настроить разбор коллекций следующим образом (для функции `test`):

```
<param name="advanced-result-selector.test.s" param="Products/Product"/>
<param name="advanced-result-selector-param.test.s.serviceId"
       param= "ServiceId/text()"/>
<param name="advanced-result-selector-param-title.test.s.serviceId"
       param="Номер сервиса"/>
<param name="advanced-result-selector-param.test.s.address"
       param= "Address/text()"/>
<param name="advanced-result-selector-param-title.test.s.address"
       param="Адрес"/>
<param name="advanced-result-selector-param.test.s.account"
       param= "Account/text()"/>
<param name="advanced-result-selector-param-title.test.s.account"
       param="Номер счета"/>

<param name="advanced-result-selector-param.test.s.title"
       param= "Address/text()"/>

<param name="advanced-result-selector-collection.test.s.meterId"
```

```
       param= «Meters/Meter/Id»/>
<param name="advanced-result-selector-collection-title.test.s.meterId"
       param="Номер счетчика»/>
<param name="advanced-result-selector-collection-
             expression.test.s.meterId" param=«text()»/>
<param name="advanced-result-selector-collection.test.s.meterName"
       param= «Meters/Meter/Name»/>
<param name="advanced-result-selector-collection-title.test.s.meterName"
       param="Название счетчика»/>
<param name="advanced-result-selector-collection-
             expression.test.s.meterName" param=«text()»/>
```

Тогда на клиента будет возвращен **«Selector»** с кодом **s**, включающий два элемента выбора с установленными надписями в атрибуте **title** (адрес), а также с набором из параметров **serviceId**, **address**, **account** и параметрами **meterId1**, **meterId2**, **meterName1**, **meterName2**.

4.6.1 ПОСЛЕДОВАТЕЛЬНЫЙ ВЫЗОВ НЕСКОЛЬКИХ ADVANCED-ФУНКЦИЙ

Доступна возможность последовательного вызова нескольких advanced-функций.

Ниже приведен пример заполнения секции для функции `test`:

```
<param name="advanced-func.test" value="func1,func2,func3"/>
```

где `test` — advanced-функция, параметры ответа которой могут использоваться в следующей вызываемой функции, `func1`, `func2` и `func3` — advanced-функции, которые будут последовательно вызваны после функции `test`. После вызова каждой функции становятся доступны ее параметры ответов, если для них определены коды ответов. Общий результат будет содержать параметры ответов всех функций.

4.7 ЗАПРОС БАЛАНСА

Данный запрос не является обязательным и настраивается, если поставщик работает по авансовой схеме и его шлюз поддерживает запрос баланса.

В случае отсутствия **balance-url**, запрос баланса выполняться не будет.

```
<param name="balance-url" param="/index.php"/>
<param name="balance-params"
       param="action=balance;login=ps;password=md5(ddwdewd)"/>

<!-- Путь к результату ответа в xml -->
<param name="balance-result-path" param="response/balance"/>
```

В качестве параметров запроса баланса указываются:

1. **balance-url** — относительный URL запроса.
2. **balance-params** — GET или POST параметры запроса через точку с запятой «;».
3. **balance-result-path** — путь к результату в XML-ответе ПУ (например, ответ **<result><balance>100.00</balance></result>**).

Если ПУ возвращает в ответе баланс в копейках, тогда в конфигурационном файле необходимо прописать параметр **balance-result-in-penny**, установив его значение равным **1**.

4.8 ЗАПРОС ОТМЕНЫ ПЛАТЕЖА

Запрос отмены платежа поддерживается реализацией **Extended** универсального шлюза.

Пример:

```
<!--URL для запроса статуса-->
<param name="cancel-request-url" param="/url/to/cansel/"/>
<param name="cancel-request-params"
       param="reqoestId=#reject-request-id#;summ=#reject-sum#"/>
<param name="cancel-request-result" param="response/cancel"/>
<param name="cancel-request-xml" param="request/cancel"/>
```

Параметры запроса отмены разделены на несколько групп:

1. URL для запроса отмены (в конфигурации необходимо указать один из трех параметров ниже):

   1) **cancel-request-url.<номер сервиса провайдера>** — URL для запроса для конкретного номера сервиса провайдера.

   2) **cancel-request-url** — URL для запроса отмены общий.

   3) **cancel-url** — URL для запроса отмены общий.

2. Параметры запроса (в конфигурации необходимо указать один из трех параметров ниже):

   1) **cancel-request-params.<номер сервиса провайдера>** — GET или POST параметры запроса для конкретного сервиса.

   2) **cancel-request-params** — GET или POST параметры запроса общие.

   3) **cancel-params** — GET или POST параметры запроса общие.

3. Путь к результату в XML-ответе ПУ (в конфигурации необходимо указать один из трех параметров ниже):

1) **cancel-request-result.<номер сервиса провайдера>** — XPath-запрос к XML-документу, если тип ответа XML, для конкретного сервиса провайдера.

2) **cancel-request-result** — общий XPath-запрос к XML-документу, если тип ответа XML.

3) **cancel-result** — общий XPath-запрос к XML-документу, если тип ответа XML.

4. Если ПУ возвращает в ответе не XML, а простой текст, тогда необходимо прописать один из параметров ниже, в которых с помощью **#result#** задать место ответа, а с помощью **#transaction#** — место транзакции (в конфигурации необходимо указать один из трех параметров ниже):

1) **cancel-request-response-text.<номер сервиса провайдер>** — регулярное выражение, выбирающее результат, если тип ответа не XML, для конкретного сервиса провайдера.

2) **cancel-request-response-text** — регулярное выражение, выбирающее результат, если тип ответа не XML.

3) **cancel-response-text** — регулярное выражение, выбирающее результат, если тип ответа не XML, для конкретного сервиса провайдера.

Пример:

```
<param name="cancel-request-response-text"
       param="#result#:#transaction#"/>
```

5. Путь к текстовому (XML) шаблону для отправки его теле запроса(в конфигурации необходимо указать один из трех параметров ниже):

1) **cancel-request-xml.<номер сервиса провайдера>** — для конкретного сервиса провайдера.

2) **cancel-request-xml** — общий путь к текстовому (XML) шаблону для отправки его теле запроса.

3) **cancel-xml** — общий путь к текстовому (XML) шаблону для отправки его теле запроса.

4.9 ЗАПРОС СТАТУСА ОТМЕНЫ ПЛАТЕЖА

Запрос статуса отмены поддерживается реализацией **Extended** универсального шлюза.

Пример:

```
<param name="cancel-status-url" param="/url/to/cansel/"/>
<param name="cancel-status-params"
       param="reqoestId=#reject-request-id#;summ=#reject-sum#"/>
<param name="cancel-status-result" param="response/status"/>
<param name="cancel-status-xml" param="request/status"/>
```

Параметры запроса статуса разделены на несколько групп.

1. URL для запроса статуса (в конфигурации необходимо указать один из трех параметров ниже):

   1) **cancel-status-url.<номер сервиса провайдера>** — URL для запроса статуса отмены для конкретного номера сервиса провайдера.

   2) **cancel-status-url** — URL для запроса статуса отмены общий.

   3) **cancel-url** — URL для запроса статуса отмены общий.

2. Параметры запроса статуса отмены (в конфигурации необходимо указать один из трех параметров ниже):

   1) **cancel-status-params.<номер сервиса провайдера>** — GET или POST параметры запроса статуса отмены для конкретного сервиса.

   2) **cancel-status-params** — GET или POST параметры запроса статуса отмены общие.

   3) **cancel-params** — GET или POST параметры запроса статуса отмены общие.

3. Путь к результату в XML-ответе ПУ (в конфигурации необходимо указать один из трех параметров ниже):

1) **cancel-status-result.<номер сервиса провайдера>** — XPath-запрос к XML-документу, если тип ответа XML, для конкретного сервиса провайдера.

2) **cancel-status-result** — общий XPath-запрос к XML-документу, если тип ответа XML.

3) **cancel-result** — общий XPath-запрос к XML-документу, если тип ответа XML.

4. Если ПУ возвращает в ответе не XML, а простой текст, тогда необходимо прописать **один из параметров ниже**, в которых с помощью **#result#** задать место ответа, а с помощью **#transaction#** — место транзакции (в конфигурации необходимо указать один из трех параметров ниже):

1) **cancel-status-response-text.<номер сервиса провайдер>** — регулярное выражение, выбирающее результат, если тип ответа не XML, для конкретного сервиса провайдера.

2) **cancel-status-response-text** — регулярное выражение, выбирающее результат, если тип ответа не XML.

3) **cancel-status-text** — регулярное выражение, выбирающее результат, если тип ответа не XML, для конкретного сервиса провайдера.

Пример:

```
<param name="cancel-status-response-text"
       param="#result#:#transaction#"/>
```

5. Путь к текстовому (XML) шаблону для отправки его теле запроса(в конфигурации необходимо указать один из трех параметров ниже):

1) **cancel-status-xml.<номер сервиса провайдера>** — для конкретного сервиса провайдера.

2) **cancel-status-xml** — общий путь к текстовому (XML) шаблону для отправки его теле запроса.

3) **cancel-xml** — общий путь к текстовому (XML) шаблону для отправки его теле запроса.

4.10 ОБЩИЕ ПАРАМЕТРЫ ШЛЮЗА

К общим параметрам шлюза относятся:

1. **`timeout`** — время таймаута отправки платежей ПУ в секундах.
2. **`date-format`** — формат даты и времени в запросах (подробнее по адресу http://download.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html).
3. **`method`** — метод отправки запроса: GET, POST, XML. Для каждой из категорий запросов возможно переопределять свой метод отправки используя префикс категории (например, `check-method` или `pay-method`), таким образом можно например запрос проверки отправлять GET-методом, а запрос платежа POST-методом.
4. **`encode`** — кодировать или нет параметры запроса URLEncode (по умолчанию **`true`**).
5. **`response-charset`** — кодировка ответа.
6. **`Response-type`** — тип ответа ПУ: **`xml`** (по умолчанию) или **`text`**.
7. **`time-format`** — формат времени в запросах (подробнее по адресу http://download.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html).
8. **`timezone`** — опционально временная зона, в которой форматировать время.
9. **`date-timezone`** — опционально временная зона, в которой форматировать дату.

4.11 ДОСТУПНЫЕ ПЕРЕМЕННЫЕ И ФУНКЦИИ В ПАРАМЕТРАХ ЗАПРОСА

Для указания параметров платежа в параметрах запроса (методом GET или POST), используется конструкция #**название параметра**#, например:

```
<param name="pay-params"
param="action=payment;receipt=#id#;date=#date#;number=#id1#;amount=#realS
um#"/>
```

Доступны следующие параметры:

1. **id** — ID платежа.
2. **date** — дата и время в формате date-format.
3. **realSum** — сумма в рублях.
4. **sum** — сумма в копейках.
5. **point** — номер точки.
6. **check** — номер чека.
7. **sumIn** — сумма вложенная в копейках.
8. **realSumIn** — сумма вложенная в рублях.
9. **comm** — сумма комиссии в копейках.
10. **realComm** — сумма комиссии в рублях.
11. **comm-used** — есть ли комиссия в платеже (1 или 0).
12. **time** — время в формате time-format.
13. **provider-service** — номер сервиса ПУ.
14. **provider-trans** — номер транзакции ПУ.
15. **point-trans** — номер операции на точке.
16. **id1,id2,...** — данные из формы.

17. **date_process** — дата "time_process". Для использования необходимо включить параметры **timezone**, **date-timezone** в конфигурационном файле (раздел 4.10).
18. **time_process** — время "time_process". Для использования необходимо включить параметры **timezone**, **date-timezone** в конфигурационном файле (раздел 4.10).
19. **date_point** — дата "pointDate". Для использования необходимо включить параметры **timezone**, **date-timezone** в конфигурационном файле (раздел 4.10).
20. **time_point** — время "pointDate. Для использования необходимо включить параметры **timezone**, **date-timezone** в конфигурационном файле (раздел 4.10).

В дополнение к параметрам возможно использовать функции, например:

```
<param name="pay-params"
param="action=payment;receipt=#id#;date=#date#;number=#id1#;amount=#realS
um#;lower(md5(#id##id1#somepassword))"/>
```

Доступны следующие функции:

1. `lower` — приведение строки к нижнему регистру.
2. `upper` — приведение строки к верхнему регистру.
3. `md5` — получение md5 hash от строки.
4. `sha1` — получение sha1 hash от строки.
5. `base64` — получение base64 от строки.
6. `signBase64` — получение ЭЦП от строки в формате base64.
7. `signHex` — получение ЭЦП от строки в формате Hex.

В параметрах запроса возможно указывать параметры первой банковской операции в списке банковских операций. При этом используется аналогичная конструкция: **#название параметра#**. Подробное описание доступных параметров приведено в разделе 4.12.1.

4.12 ОТПРАВКА ЗАПРОСОВ В XML-ФОРМАТЕ В ТЕЛЕ ЗАПРОСА

Для отправки запросов в XML-формате:

1. Укажите тип запроса **`method=XML`**.
2. Для используемых типов запросов укажите пути к файлам с шаблонами запросов:

```
<param name="check-request-xml"
param="/home/gates/configs/uni/check.xml"/>
<param name="pay-request-xml" param="/home/gates/configs/uni/pay.xml"/>
<param name="status-request-xml"
param="/home/gates/configs/uni/status.xml"/>
<param name="confirm-request-xml"
param="/home/gates/configs/uni/confirm.xml"/>
<param name="advanced-request-xml.test"
param="/home/gates/configs/uni/test.xml"/>
```

Сами шаблоны представляют XML-запросы как есть, с вставками данных из платежа и параметров шлюза, с помощью velocity (подробнее в руководстве пользователя velocity по адресу http://velocity.apache.org/engine/devel/user-guide.html).

Пример шаблона платежа:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv=
                  "http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="1"
                 xmlns:wsse="http://docs.oasis-open.org/wss/
                             2004/01/oasis-200401-wss-wssecurity-secext-
                             1.0.xsd">
   <wsse:UsernameToken wsu:Id="UsernameToken"
                 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
                           oasis-200401-wss-wssecurity-utility-1.0.xsd">
    <wsse:Username>$gateParams.api-username</wsse:Username>
    <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/
```

```
                        oasis-200401-wss-username-token-profile-
                        1.0#PasswordText">$gateParams.api-
                        password</wsse:Password>
   </wsse:UsernameToken>
  </wsse:Security>
 </soapenv:Header>
 <soapenv:Body>
  <TransferRequest xmlns="http://www.moneta.ru/schemas/messages.xsd">
   <payer>$gateParams.service-account-id</payer>
   <payee>$payment.account.identifier</payee>
   <amount>$numberFormat.format($payment.realSumOutcome)</amount>
   <isPayerAmount>true</isPayerAmount>
   <paymentPassword>
    $gateParams.service-payment-password
   </paymentPassword>
   <clientTransaction>$payment.id</clientTransaction>
  </TransferRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

В шаблоне доступны следующие объекты:

1. **payment** — объект платежа.
2. **dateFormat** — форматтер даты (пример: **`$dateFormat.format($payment.date,"dd.MM.yyyy HH:mm:ss")`**).
3. **numberFormat** — форматтер суммы (пример: **`$numberFormat.format($payment.realSumOutcome)`**.
4. **now** — текущее время (пример: **`$dateFormat.format($now,"dd.MM.yyyy HH:mm:ss")`**).
5. **gateParams** — параметры шлюза из конфигурационного файла, доступ $gateParams.<имя параметра>, (пример $gateParams.login).
6. **integerFormat** — целочисленный формат.
7. **hash** — хэш-генератор, доступны методы `calcMD5`, `calcSHA1`, `base64`.

Пример:

```
#set($forSign = $payment.account.identifier+$gateParams.get("signPwd"))
```

```
...
$hash.calcMD5($forSign).toLowerCase()
```

8. **signer** — генератор ЭЦП, доступны методы signBase64, signHex.
9. **String** — доступны методы Java-класса String.
10. **Double** — доступны методы Java-класса Double.
11. **Integer** —доступны методы Java-класса Integer.
12. **renderValues** — карта доступных параметров из рендеров шлюза для сервиса платежа.
13. Все дополнительные параметры объявленные как velocity-переменные в конфигурационном файле шлюза доступны по их названию.

В **payment** доступны следующие атрибуты:

1. **`$payment.id`** — ID платежа.
2. **`$payment.sumOutcome`** — сумма в копейках.
3. **`$payment.realSumOutcome`** — сумма в рублях.
4. **`$payment.sumIncome`** — сумма вложенная в копейках.
5. **`$payment.sumComm`** — сумма комиссии в копейках.
6. **`$payment.sumProv`** — сумма в валюте поставщика в копейках (если валюта проведения отличается от валюты приема/сервера).
7. **`$payment.date`** — дата платежа.
8. **`$payment.point.id`** — ID точки.
9. **`$payment.check`** — номер чека.
10. **`$payment.account.externalService`** — номер услуги ПУ.
11. **`$payment.account.identifier`** — **`id1`** из формы.
12. **`$payment.account.identifier2`** — **`id2`** из формы.
13. **`$payment.account.attributes.<название атрибута>`** — другие атрибуты из формы.

14. **$payment.providerTrans** — номер транзакции поставщика.
15. **$payment.providerDate** — дата обработки операции.
16. **$payment.pointTrans** — номер операции на точке.
17. **$payment.pointTrans** — номер операции на точке.
18. **$payment.order** — параметры товара или услуги.

Пример обращения к атрибуту $payment.order в Velocity-шаблоне:

```
Order: $payment.order
Order items:
#if($payment.order)
#foreach($item in $payment.order.items)
------------------------------------------------
code: $item.code
name: $item.name
price: $item.price
amount: $item.amount
fiscalSubject: $item.fiscalSubject
fiscalSubject.id: $item.fiscalSubject.id
fiscalSubject.name: $item.fiscalSubject.name
fiscalSubject.printName: $item.fiscalSubject.printName
fiscalSubject.isDefault: $item.fiscalSubject.defaultSubj
taxRate: $item.taxRate
taxRate.id: $item.taxRate.id
taxRate.code: $item.taxRate.code
taxRate.name: $item.taxRate.name
taxRate.rate: $item.taxRate.rate
taxRate.type: $item.taxRate.type
#end
#end
```

Пример вывода параметров товара:

```
Order: Order{items=[OrderItem{code='123FD', name='Чебурек вчерашний',
price=5000, amount=1000, fiscalSubject=FiscalSubject{id=1,
name='О реализуемом товаре, за исключением подакцизного товара
(наименование и иные сведения, описывающие товар)',
printName='Товар', defaultSubj=false}, taxRate=TaxRate{id=1,
code='VAT_18', name='НДС 20%', rate=0.20000, type='VAT'}},
```

```
OrderItem{code='228FD', name='Эспумизан большой', price=15000,
amount=10000, fiscalSubject=FiscalSubject{id=2,
name = 'О реализуемом подакцизном товаре (наименование и иные сведения,
описывающие товар)', printName='Подакцизный товар', defaultSubj=false},
taxRate=TaxRate{id=2, code='VAT_10', name='НДС 10%', rate=0.10000,
type='VAT'}}]}

Order items:
-----------------------------------------------------
code: 123FD
name: Чебурек вчерашний
price: 5000
amount: 1000
fiscalSubject: FiscalSubject{id=1, name='О реализуемом товаре, за
исключением подакцизного товара (наименование и иные сведения,
описывающие товар)', printName='Товар', defaultSubj=false}
fiscalSubject.id: 1
fiscalSubject.name: О реализуемом товаре, за исключением подакцизного
товара (наименование и иные сведения, описывающие товар)
fiscalSubject.printName: Товар
fiscalSubject.isDefault: false
taxRate: TaxRate{id=1, code='VAT_18', name='НДС 20%', rate=0.20000,
type='VAT'}
taxRate.id: 1
taxRate.code: VAT_18
taxRate.name: НДС 20%
taxRate.rate: 0.20000
taxRate.type: VAT
-----------------------------------------------------
code: 228FD
name: Эспумизан большой
price: 15000
amount: 10000
fiscalSubject: FiscalSubject{id=2, name='О реализуемом подакцизном товаре
(наименование и иные сведения, описывающие товар)',
printName='Подакцизный товар', defaultSubj=false}
fiscalSubject.id: 2
fiscalSubject.name: О реализуемом подакцизном товаре (наименование и иные
сведения, описывающие товар)
fiscalSubject.printName: Подакцизный товар
fiscalSubject.isDefault: false
```

```
taxRate: TaxRate{id=2, code='VAT_10', name='НДС 10%', rate=0.10000,
type='VAT'}
taxRate.id: 2
taxRate.code: VAT_10
taxRate.name: НДС 10%
taxRate.rate: 0.10000
taxRate.type: VAT
```

4.12.1 АТРИБУТЫ БАНКОВСКИХ ОПЕРАЦИЙ

Доступна возможность использовать атрибуты банковских операций для дальнейшей передачи этих данных ПУ.

В объект платежа **payment** добавлен список банковских операций, связанных с данным платежом. Методы, позволяющие получить доступ к списку банковских операций, приведены в Таблице 1.

Таблица 1 — Методы, использующиеся для доступа к списку банковских операций

Название	Описание
$payment.getBankOperations()	Возвращает список банковских операций, связанных с платежом
$payment.getFirstBankOperation()	Возвращает первую банковскую операцию из списка

Также добавлен объект **bo**, позволяющий получить доступ к первой банковской операции. Обращение к данному объекту аналогично вызову метода $payment.getFirstBankOperation().

В **bo** доступны следующие атрибуты:

1. **`$bo.id`** — ID операции.

2. **$bo.point** — объект, содержащий информацию о точке, с которой была проведена операция.
3. **$bo.date** — время регистрации платежа на сервере.
4. **$bo.paymentToolId** — ID платёжного инструмента.
5. **$bo.sumIncome** — сумма вложенной наличности в копейках.
6. **$bo.sumOutcome** — сумма платежа, зачисленная в копейках.
7. **$bo.sumComm** — сумма комиссии в копейках.
8. **$bo.currency** — валюта операции.
9. **$bo.providerId** — ID провайдера.
10. **$bo.rrn** (Retrieval Reference Number) — номер транзакции провайдера.
11. **$bo.authCode** — код авторизации.
12. **$bo.pma** (Payment Method Account) — первичный идентификатор объекта, по которому совершена транзакция. Например, номер карты или кошелька.
13. **$bo.token** — присвоенный провайдером уникальный идентификатор объекта, по которому совершена транзакция.
14. **$bo.session** — ID сессии для списания на терминале.
15. **$bo.merchantId** — ID мерчанта в унифицированной системе эквайринга.
16. **$bo.attributes** — список атрибутов операции.
17. **$bo.terminalCode** — код терминала.
18. **$bo.merchantCode** — код мерчанта.
19. **$bo.cardName** — название карты. Например, Visa, Mastercard и т.п.
20. **$bo.cardType** — тип карты.

В **bo.point** доступны следующие атрибуты:

1. **$bo.point.id** — ID точки.
2. **$bo.point.name** — наименование точки.

3. **$bo.point.city** — город, в котором расположена точка.
4. **$bo.point.address** — адрес точки.
5. **$bo.point.code** — код точки.
6. **$bo.point.pointType** — тип точки.

Атрибуты первой банковской операции, доступные в параметрах запросов:

1. **ps_id** — ID операции.
2. **ps_date** — дата операции.
3. **ps_time** — время операции.
4. **payment_tool_id** — ID платёжного инструмента.
5. **ps_point** — ID точки.
6. **ps_sum** — сумма платежа, зачисленная в копейках.
7. **ps_sumrub** — сумма платежа, зачисленная в рублях.
8. **ps_sumIn** — сумма вложенной наличности в копейках.
9. **ps_realSumIn** — сумма вложенной наличности в рублях.
10. **ps_comm** — сумма комиссии в копейках.
11. **ps_realComm** — сумма комиссии в рублях.
12. **ps_currency** — валюта операции.
13. **ps_rrn** — (Retrieval Reference Number) — номер транзакции провайдера.
14. **ps_auth_code** — код авторизации.
15. **ps_pma** — (Payment Method Account) — первичный идентификатор объекта, по которому совершена транзакция. Например, номер карты или кошелька.
16. **ps_token** — присвоенный провайдером уникальный идентификатор объекта, по которому совершена транзакция.
17. **ps_session** — ID сессии для списания на терминале.

18. **ps_merchant_id** — ID мерчанта в унифицированной системе эквайринга.
19. **ps_terminal_code** — код терминала.
20. **ps_merchant_code** — код мерчанта.
21. **ps_card_name** — название карты. Например, Visa, Mastercard и т.п.
22. **ps_card_type** — тип карты.
23. **ps_имя_атрибута** — подстановка произвольного атрибута банковской операции, который используется в протоколе. Например, **ps_fio** — ФИО, **ps_dolg** — задолженность.

4.13 ЭКРАНИРОВАНИЕ СПЕЦСИМВОЛОВ В ЗАПРОСАХ

Экранировать спецсимволы в Velocity-шаблоне конфигурационных файлов шлюзов позволяет класс `StringEscapeUtils`.

Для запросов JSON-формата пропишите в теле запроса метод:

```
$StringEscapeUtils.escapeJson($val)
```

Для запросов XML-формата пропишите в теле запроса метод:

```
$StringEscapeUtils.escapeXml10($val)
```

Пример до использования метода:

```
"бирка\ключ" 'тест' новый/тест ""всё сломал"" ''еще раз'' тут      таб
```

Пример после использования метода:

```
\"\u0431\u0438\u0440\u043A\u0430\\\u043A\u043B\u044E\u0447\" '\u0442\
u0435\u0441\u0442' \u043D\u043E\u0432\u044B\u0439\/\u0442\u0435\u0441\
u0442 \"\"\u0432\u0441\u0451 \u0441\u043B\u043E\u043C\u0430\u043B\"\" ''\
u0435\u0449\u0435 \u0440\u0430\u0437'' \u0442\u0443\u0442\t\u0442\u0430\
u0431
```

Подробнее о применении класса `StringEscapeUtils` по адресу: https://javadoc.io/static/org.apache.commons/commons-text/1.8/org/apache/commons/text/StringEscapeUtils.html.

4.14 ИСПОЛЬЗОВАНИЕ HTTP-ЗАГОЛОВКОВ

Бывают ситуации, когда ПУ требует передачи в HTTP-заголовках определенных значений, для этих целей, для каждой категории запросов возможно определять секцию `headers` с префиксом, определяющим тип запроса, например:

```
<param name="check-headers" param="Content-type=text/xml,Sign=signBase64(#id##id1#)"/>

<param name="pay-headers" param="Content-type=text/xml,Sign=signBase64(#id##id1#)"/>
```

В данном случае будут добавлены два HTTP-заголовка к запросам проверки и проведения платежа: Content-type и Sign.

В качестве значения заголовков можно использовать все параметры и функции из раздела 4.11.

4.15 ВЫЧИСЛЕНИЕ ПАРАМЕТРОВ НА ЯЗЫКЕ JAVASCRIPT

Функционал универсального шлюза позволяет определять выражения на языке JavaScript, которые будут обрабатываться при каждом запросе и результат выполнения которых будет доступен в параметрах проведения платежа, как в GET/POST-запросах через **#param_name#**, так и в XML-запросах через **$param_name**.

Определение JavaScript-параметров делается следующим образом:

```
<param name="js_hash" param="hash.calcMD5(''+(payment.getSumIncome()
+payment.getSumOutcome())/100.0)"/>
```

В данном случае в контексте запроса будет доступен параметр **hash**, как результат выполнения данного скрипта.

В скрипте доступны следующие объекты:

1. **payment** — объект платежа.
2. **hash** — хэш генератор.
3. **nf** — форматтер суммы (пример: nf.format(payment.getSumOutcome())).
4. **sdf** — форматтер даты (пример:sdf.format(payment.getDate(),"dd.MM.yyyy HH:mm:ss").

4.16 ОПРЕДЕЛЕНИЕ ДОПОЛНИТЕЛЬНЫХ VELOCITY-ПАРАМЕТРОВ

Функционал универсального шлюза позволяет определять дополнительные velocity-шаблоны, которые будут обрабатываться при каждом запросе и результат выполнения которых будет доступен в параметрах проведения платежа, как в GET/POST-запросах через **#velocity_param_name#**, так и в XML-запросах через **$velocity_param_name**.

Определение Velocity-параметров делается следующим образом:

```
<param name="velocity_forSign"
       param="/home/gates/configs/gate/forSign.xml"/>
```

В данном случае в контексте запроса будет доступен параметр **forSign**, как результат обработки velocity-шаблона по указанному пути в файловой системе.

В шаблоне доступны все те же объекты и переменные, описанные в разделе 4.12.

Данный механизм удобно использовать для вычисления ЭЦП (раздел 4.18), когда строку для подписи мы определяем во внешнем velocity-шаблоне, и потом в самом запросе используем вызов функции от конкретной переменной:

```
<param name="pay-params"
     param="id=#id#;account=#id1#;sum=#sum#;sign=signBase64(#forSign#)"/>
```

4.17 ИСПОЛЬЗОВАНИЕ РЕНДЕРОВ

Функционал универсального шлюза позволяет использовать механизм рендеров, предполагающий определение переменных и их значений в параметрах сервиса провайдера в кабинете.

Для использования рендеров в конфигурационном файле шлюза необходимо указать параметр:

```
<param name="render-loader" param="true">
```

По умолчанию значение переменной — **`false`**.

В универсальном шлюзе с использованием рендеров мы можем переопределять URL для каждой категории запросов, например, `check-url` или `pay-url`, а также параметры запроса, путь для кода ответа или путь к velocity-шаблону.

В случае указания данных переменных в параметрах сервиса провайдера, именно они будут использоваться при проведении платежей по данному сервису.

Дополнительно возможно определять любые переменные, которые будут доступны в контексте платежа:

```
render.check-url=/check263.php
render.pay-url=/pay263.php
render.check-params=id=#id#;account=#id1#
render.pay-params=id=#id#;account=#id1#;sum=#sum#;param=#param1#
render.param1=#id1##id2#
```

В данном случае, будут использованы указанные URL для проведения платежа, вместо определенных URL в конфигурационном файле универсального шлюза, будут использовать определенные параметры проверки и платежа, также будет доступен параметр **`param1`** в контексте всех запросов, как GET/POST, так и XML-запросов.

Поток подгрузки настроек рендеров выполняется при старте шлюза, а также каждые 10 минут работы шлюза.

4.18 ИСПОЛЬЗОВАНИЕ ЭЛЕКТРОННО-ЦИФРОВОЙ ПОДПИСИ ЗАПРОСОВ

В случае необходимости использования электронно-цифровой подписи запросов, требуется определить местонахождение и пароль закрытого ключа, алгоритм подписи и кодировку, в которой получать байты из строки для подписи:

```
<param name="signer-private-key"
param="/home/gates/configs/gate/private.pem"/>
<param name="signer-private-key-password" param="password"/>

<param name="signer-public-key"
param="/home/gates/configs/gate/public.pem"/>

<param name="signer-algoritm" param="SHA1WithRSA"/>
<param name="signer-charset" param="UTF8"/>
```

Ключи должны быть в PEM-формате, в случае если закрытый ключ имеет пароль, его необходимо указать.

Опциональными являются указание алгоритма подписи и кодировка. По умолчанию они соответственно равны SHA1withRSA и UTF8.

После определения настроек ЭЦП, в контексте GET/POST запросов можно использовать функции `signBase64` и `signHex`, а в XML-запросах будет доступен объект **signer** (раздел 4.12).

Внимание!
Реализация **Simple** не поддерживает возможность использования электронно-цифровой подписи запросов.

4.19 ИЗВЛЕЧЕНИЕ АТРИБУТОВ ИЗ ТЕКСТОВЫХ ОТВЕТОВ ПУ НА ЗАПРОСЫ ПРОВЕРКИ НОМЕРА И ПРОВЕДЕНИЯ ПЛАТЕЖА

Поддерживается разбор ответов и извлечения атрибутов из текстовых ответов ПУ с помощью регулярных выражений.

Для использования данного функционала в конфигурационном файле шлюза для параметра **response-type**, определяющего тип ответа ПУ, необходимо указать значение **regex**. В общем случае задание будет выглядеть следующим образом:

```
<param name="response-type" param="regex"/>
```

Для конкретного типа запроса синтаксис будет аналогичным. Например, для запроса проверки следует указать:

```
<param name="check-response-type" param="regex"/>
```

Названия параметров типа **regex** в конфигурационном файле аналогичны названиям параметров типа XML и JSON, но для их значений вместо **XPATH** и **JSONPath** необходимо указать регулярное выражение.

Ниже перечислены поддерживаемые виды регулярных выражений в порядке понижения приоритета:

1. Регулярное выражение содержит именованную группу `val`. В качестве значения используется строка, соответствующая данной группе. При этом для группировки возможно применять следующие типы групп:

   1) Нумерованные захватывающие группы. Используются в том случае, если необходимо последующее извлечение содержимого группы.

   Пример:

   ```
   (Баланс: )(?<val>[\d.]+)( руб)
   ```

   2) Не захватывающие группы. Применяется в том случае, если группа используется для проверки, соответствует ли строка регулярному выражению.

Пример:

```
Баланс: (?<val>[\d.]+) руб
```

3) Предпросмотр. Будет использоваться фрагмент, находящийся перед найденной строкой, при этом найденная строка не будет учитываться.

Пример:

```
(?:Баланс: )(?<val>[\d.]+)(?: руб)
```

4) Постпросмотр. Будет использоваться фрагмент, находящийся после найденной строки, при этом найденная строка не будет учитываться.

Пример:

```
(?<=Баланс: )(?<val>[\d.]+)(?= руб)
```

2. Регулярное выражение содержит нумерованные группы. В качестве значения используется строка, соответствующая первой группе. При этом для группировки возможно применять не захватывающие группы, предпросмотр и постпросмотр.

Примеры:

```
Баланс: ([\d.]+) руб
(?:Баланс: )([\d.]+)(?: руб)
(?<=Баланс: )([\d.]+)(?= руб)
```

3. Регулярное выражение не содержит нумерованные группы. В качестве значения используется вся строка, соответствующая регулярному выражению. При этом для группировки возможно применять предпросмотр и постпросмотр.

Примеры:

```
(?:Баланс: )([\d.]+)(?: руб)
(?<=Баланс: )[\d.]+(?= руб)
```

Пример настройки запроса проверки с использованием регулярных выражений, содержащих нумерованные группы:

```
<param name="check-url" param="/verify"/>
<param name="check-params" param="account=#id1#"/>
<param name="check-response-type" param="regex"/>
<param name="check-result-path" param="error: (\d+)"/>
```

```
<param name="check-result-param.fio" param="Имя: (.+)"/>
<param name="check-result-param.debt" param="Баланс: ([\d.-]+)"/>
<param name="check-result-param.missing" param="Такого нет: (.+)"/>
```

Пример ответа на запрос проверки:

```
error: 0
Имя: Иванов Иван Иванович
Баланс: -123.45 руб
```

Пример настройки запроса проведения:

```
<param name="pay-url" param="/pay"/>
<param name="pay-params" param="account=#id1#"/>
<param name="pay-response-type" param="regex"/>
<param name="pay-result-path" param="error: (\d+)"/>
<param name="pay-result-param.debt" param="Баланс: ([\d.-]+)"/>
<param name="pay-result-param.pin" param="pin: ([^\s]+)"/>
<param name="pay-timeprocess-path" param="Время: (\d\d\d\d-\d\d-\d\dT\d\
d:\d\d:\d\d\.\d\d\d)"/>
<param name="pay-transaction-path" param="Транзакция: ([\d.-]+)"/>
<param name="response-date-format" param="yyyy-MM-dd'T'HH:mm:ss.SSS"/>
```

Пример ответа на запрос проведения:

```
error: 0
Баланс: -123.45 руб
Время: 2019-08-19T09:01:02.345
Транзакция: 12345
pin: abc-999
```

4.20 ОТВЕТЫ НА ЗАПРОСЫ В ФОРМАТЕ JSON

Ответы на различные запросы возможно получать в формате JSON. Для использования данного функционала в конфигурационном файле шлюза для параметра **response-type**, определяющего тип ответа ПУ, необходимо указать значение **json**. В общем случае задание будет выглядеть следующим образом:

```
<param name="response-type" param="json"/>
```

Для конкретного типа запроса синтаксис будет аналогичным. Например, чтобы на запросы проверки возвращался ответ в формате JSON, в конфигурационном файле следует указать:

```
<param name="check-response-type" param="json"/>
```

Для поиска кодов значения, транзакции и произвольных параметров используются те же пути, что и для ответов в формате XML, но при этом синтаксис указания данных путей отличается. Для того, чтобы извлечь данные из ответа в формате JSON, используется язык запросов JSONPath. В этом случае, чтобы вернуть значение параметра, необходимо указать путь от корня JSON до нужного элемента.

Пример указания пути к результату ответа в формате JSON:

```
<param name="check-result-path" param="$.status"/>
```

Символ **$** является корневым элементом ответа в формате JSON, **status** — параметр (дочерний элемент), в котором содержаться нужные данные. Более подробно синтаксис языка запросов JSONPath описан на странице https://github.com/json-path/JsonPath.

Ниже приведен пример возврата двух параметров **date** и **sum** посредством JSONPath:

```
<param name="check-result-param.date" param="$.date"/>
<param name="check-result-param-title.date" param="Дата"/>
<param name="check-result-param.sum" param="$.sum"/>
<param name="check-result-param-title.sum" param="Сумма к оплате"/>
```

В данном случае вернутся два параметра со значениями из ответа ПУ, а также установленными **title**.

4.21 ПЕРЕДАЧА ПАРАМЕТРОВ ПЛАТЕЖА В URL-ЗАПРОСЕ

Для того, чтобы осуществить подстановку параметров платежа в URL-запрос, в конфигурационном файле шлюза в секции <params> </params> необходимо указать параметр **replace-url-params** со значением **true**:

```
replace-url-params=true
```

Возможно использовать только для запросов вида **check-url**, **pay-url**, **status-url**, **advanced-url**.

Пример:

```
<param name="pay-url" param="/test/#id#"/>
```

где **pay-url** — относительный URL запроса проведения платежа, /test/ — произвольная часть URL, **id** — номер транзакции.

Также в качестве параметра платежа могут использоваться данные из формы, введенные клиентом.

Пример:

```
<param name="check-url" param="/test/#id1#"/>
```

где **check-url** — относительный URL запроса проверки номера абонента, /test/ — произвольная часть URL, **id1** — номер абонента.

4.22 ОТКЛЮЧЕНИЕ ОФЛАЙН-ПРОВЕРКИ РЕКВИЗИТОВ ПЛАТЕЖА

В случае стандартного проведения платежа кроме офлайн-проверки валидности его реквизитов может применяться онлайн-проверка. Онлайн-проверка необходима для проверки реквизитов платежа и/или получения данных от поставщика услуги до внесения денежных средств.

Выполнение онлайн-проверки позволяет:

1. Уменьшить количество ошибок, возникающих при проведении платежа. Например, предварительная проверка номера, введенного клиентом при оплате.
2. Реализовать бизнес-логику конкретного решения. Например, если необходимо получить номер сессии для последующих запросов или реквизиты поставщика от агрегатора.

Обе проверки совершаются перед запросом проведения платежа, однако онлайн-проверка происходит в режиме реального времени, а офлайн — в тот момент, когда платеж достигнет начала очереди платежей.

При этом в онлайн- и офлайн-режимах может происходить проверка одних и тех же реквизитов платежа. Во избежание повторных проверок, а также в других случаях, предусмотренных бизнес-схемой проведения платежа, существует возможность отключить офлайн-проверку реквизитов на стороне сервера.

Для того, чтобы отключить офлайн-проверку реквизитов платежа, в конфигурационном файле шлюза в секции задания его параметров укажите параметр **`only-online-verify`** со значением **`true`**:

```
<param name="only-online-verify" param="true"/>
```

4.23 ПОЛУЧЕНИЕ КОМИССИИ ОТ ПОСТАВЩИКА

Для получения от поставщика информации о комиссии и ее настройках используйте в запросе усовершенствованного обработчика параметры:

1. **advanced-result-fee-sum.<func>** — путь до значения комиссии для функции `func`.
2. **advanced-result-fee-type.<func>** — тип комиссии для функции `func`. Возможные значения: 0 — в рублях, 1 — в копейках. Если не задано, то 0.
3. **advanced-result-fee-name.<func>** — название комиссии для отображения в кабинете для функции `func`. Параметр является опциональным.
4. **advanced-result-fee-percent.<func>** — процент комиссии для отображения клиенту на экране терминала для функции `func`. Параметр является опциональным.

Для получения от поставщика настроек комиссии задайте один из двух следующих параметров:

1. **advanced-result-client-fee-percent.<func>** — путь до процента комиссии для функции `func`.
2. **advanced-result-client-fee-fixed.<func>** — путь до фиксированного значения комиссии для функции `func`.

Параметры запроса для получения от поставщика настроек комиссии:

1. **advanced-result-client-fee-type.<func>** — тип фиксированного значения комиссии в ответе для функции `func`. Возможные значения: 0 — в рублях, 1 — в копейках. Если не задано, то 0.
2. **advanced-result-client-fee-min.<func>** — путь до минимального значения комиссии для функции `func`.

3. **advanced-result-client-fee-max.<func>** — путь до максимального значения комиссии для функции `func`.

4.24 СОХРАНЕНИЕ РЕЗУЛЬТАТА ВЫПОЛНЕНИЯ ЗАПРОСА ПРОВЕДЕНИЯ ПЛАТЕЖА

Параметр **separate-confirm** позволяет сохранить в БД результат выполнения запроса проведения платежа сразу после его успешного выполнения, но перед запросом подтверждения платежа. Для того, чтобы задать параметр, добавьте в конфигурационный файл шлюза строку:

```
<param name="separate-confirm" param="true"/>
```

Если параметр не задан, то результат запроса проведения сохранится только после выполнения запроса подтверждения платежа.

Например, если при сохранении ID операции поставщика в запросе проведения и его последующей передаче в запрос подтверждения платежа шлюзы будут перезагружены, то атрибуты платежа не будут сохранены.