Paylogic

Протокол взаимодействия между поставщиком услуги и Процессинговым Центром Pay-logic

Руководство разработчика

АННОТАЦИЯ

Содержит описание и техническую спецификацию протокола взаимодействия для поставщика услуги, не имеющего собственного протокола

Версия руководства: 2.1.1

Руководство актуально для программного обеспечения «Процессинговый центр Pay-logic» версий 4.5.x

ООО «Софт-Лоджик»

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

Тел: (3852) 72-27-27

ОГЛАВЛЕНИЕ

ИСТОРИЯ ИЗМЕНЕНИЙ

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 1.0

Дата публикации: 21.04.2010.

Изменение	Раздел
Общие улучшения в документации:	
Документ создан	-

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 2.0

Дата публикации: 29.03.2015.

Изменение	Раздел
Общие улучшения в документации:	
Переоформление	-

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 2.1

Дата публикации: 27.08.2015.

Изменение	Раздел
Общие улучшения в документации:	
Переоформление	-

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 2.1.1

Дата публикации: 21.07.2017.

Изменение	Раздел
Общие улучшения в документации:	
Переоформление	-

ИЗМЕНЕНИЯ В ВЕРСИИ ДОКУМЕНТА 2.1.2

Дата публикации: 30.03.2018.

Изменение	Раздел
Общие улучшения в документации:	
Термин «электронная цифровая подпись» заменен на термин «электронная подпись»	-

1 ОСНОВНЫЕ ТЕРМИНЫ

Абонент — физическое или юридическое лицо, заключившее с Поставщиком услуг Абонентский договор.

Номер абонента — индивидуальный телефонный номер абонента, идентифицирующий его в системе Поставщика услуг.

Платеж — здесь: платеж, совершенный с помощью следующих платежных инструментов — наличные или кредитная карта, в пользу абонента.

Платежная система — юридическое лицо предоставляющее услуги процессинга, с которым заключают договора агенты.

Плательщик — здесь: конкретное физическое лицо, совершившее платеж.

Поставщик услуг — юридическое лицо предоставляющее услуги, с которым заключают договора абоненты.

Сеть приема платежей — комплекс аппаратно-технических средств, используемый для информационного взаимодействия Участников.

Таймаут — здесь: время ожидания.

Участник — юридическое лицо, подписавшее «Договор участия в системе приема платежей».

2 ОСНОВНЫЕ СОКРАЩЕНИЯ

АС — автоматизированная система.

ПУ — здесь: Поставщик услуги

ПЦ — здесь: Процессинговый центр.

ОТТ — общие технические требования. Данный документ.

СПП — сеть приема платежей.

ЭП — электронная подпись.

3 ПРЕДПОЛАГАЕМАЯ АУДИТОРИЯ

Руководство предназначено для использования в качестве спецификации для разработки протокола взаимодействия со стороны поставщика услуги.

Документ адресован IT-специалистам (программистам и администраторам), обслуживающим информационную систему поставщика услуг, знакомым с технологиями разработки ПО, языками программирования, основам работы с XML и сетевым взаимодействием по HTTP/HTTPS, владеющим основами криптографии.

Необходимо знание языков программирования и соответствующих библиотек для Java или PHP.

4 ВВЕДЕНИЕ

Документ предназначен для поставщиков услуг, не имеющих собственного протокола взаимодействия, но желающих разработать и внедрить протокол для онлайнового взаимодействия. Документ адресован разработчикам АС ПУ:

1. Для проведения доработки (разработки) программного обеспечения для АС ПУ, разработки технических заданий на доработку (разработку) соответствующих систем, обеспечивающих возможность подключения к ПУ.
2. Для оценки качества проведенных доработок и соответствия доработанной системы требованиям ОТТ во время комплексных испытаний. Реализация требований данного документа позволит подключить ПУ к СПП и обеспечить прием платежей абонентов через каналы СПП.

5 ОБЩИЕ ПОЛОЖЕНИЯ

5.1 ЦЕЛЬ И НАЗНАЧЕНИЕ

Система предназначена для обеспечения приема платежей через терминалы СПП или иные каналы СПП и проведения платежей в режиме, приближенном к режиму реального времени.

Бизнес-цель создания системы — реализовать онлайновый канал получения информации о принятых транзакциях для максимально оперативного поступления информации о транзакциях, совершаемых через СПП. Реализация протокола позволит в дальнейшем подключать иные СПП, предлагая реализованный протокол в качестве инструмента для интеграции.

В отдельных случаях может возникать необходимость подключения поставщиков услуг, не имеющих собственного протокола, но имеющих техническую возможность и желание реализовать онлайновое взаимодействие. Для решения такой задачи возможно использование либо протокола Paylogic, либо протокола ОСМП.

Протокол ОСМП рассмотрен в формуляре «Протокол ОСМП для подключения поставщиков услуг к процессинговому центру Pay-logic».

Второй вариант, протокол Pay-logic Gate. Протокол предназначен для поставщиков услуг, и предполагает реализацию на стороне поставщика услуги принимающего шлюза для обмена сообщениями в соответствии со спецификацией в настоящем документе.

Для подключения по протоколу Paylogic поставщику на своей стороне необходимо реализовать протокол в составе следующих запросов:

1. Запрос баланса (balance).
2. Запрос проверки номера (verify).
3. Запрос проведения платежа (payment).
4. Запрос статуса платежа (status).

5.2 ТРЕБОВАНИЯ К АС ПУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

Сервер АС ПУ должен быть подключен к Интернет. Обращение к серверу поставщика услуги осуществляется через публичную сеть Интернет по шифрованному соединению (HTTPS). Возможно использование шифрованных каналов VPN (IPSEC или других туннелей). Уровень защиты и способ шифрования потока данных на усмотрение поставщика услуги.

Общие требования к программному обеспечению:

1. Вся передача данных происходит по шифрованному соединению.
2. Каждый платеж (транзакция) снабжается уникальным идентификатором, который обеспечивает отсутствие повторных платежей на текущий номер.
3. АС СПП (процессинга) взаимодействует с системой, поставщика услуги отправляя POST запросы на сервер ПУ, поставщик услуги принимает запросы, проверяет их на предмет целостности и корректности, обрабатывает их и отправляет ответ СПП в соответствии с протоколом.

6 ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

6.1 СЕТЕВОЕ ВЗАИМОДЕЙСТВИЕ

Сетевое взаимодействие ПУ и ПЦ должно происходить в рамках протокола https. Допускается использовать HTTP (небезопасно).

Поставщик услуги должен предоставить для СПП адрес и порт, на который будут отправляются запросы. Поставщик услуги должен прослушивать указанный порт на указанном адресе и ожидать на этом порту HTTP или HTTPS запросы с использованием метода POST. В случае Java это означает, что на указанном URL и порту должен работать сервлет, реализующий прием и обработку запросов от СПП.

В случае, когда поставщик услуги использует несколько разных подключенных к нему СПП, либо несколько точек из одной СПП, следует разделять запросы с разных точек, назначая им разные URL для отправки запросов. На стороне ПУ должна быть реализована логика, которая соотнесет на какой URL пришел запрос и какому агенту (СПП) этот запрос соответствует. Фактически, это означает, что для каждого подключения должен работать свой сервлет. Такая технология является следствием простоты протокола.

Получив пакте от СПП, ПУ проверяет его подпись, и в случае если подпись корректна — возвращает ответ.

При ответе на полученный пакет, ПУ возвращает пакет с подписью, в котором содержатся результаты обработки данных пакета. Данные должны помещаться в теле страницы, генерируемой в ответ на обращение со стороны АС ВПС.

Сервер ПУ должен иметь возможность обрабатывать параллельно несколько запросов.

Авторизация на сервере ПУ может происходить двумя способами:

1. С использованием клиентского сертификата.
2. С использованием средств Web-сервера (например, BASIC).

6.2 ЭЛЕКТРОННАЯ ПОДПИСЬ

Для гарантии подлинности отправителя и получателя используется электронная подпись запроса и ответа. Для авторизации запроса ЭП передается в формате Base64 в HTTP заголовке запроса PayLogic-Signature. Для авторизации ответа ЭП передается в формате Base64 в HTTP заголовке ответа PayLogic-Signature.

Генерация ключей осуществляется поставщиком услуги.

Для генерации ключей можно использовать любые утилиты, позволяющие генерировать RSA ключи, длину ключа выбирать равной 1024 или 2048 бит.

ПРИМЕР:

Для создания закрытого ключа используется команда genrsa:

openssl genrsa [-out file] –des3 [bits]

Команда genrsa создает секретный ключ длиной bits в формате PEM.

Для создания публичного ключа rsa на основе секретного используется команда openssl rsa. Данная команда имеет следующий формат:

openssl rsa –pubout -in filename [-out file],

где filename — имя файла с закрытым ключом.

Примеры создания и проверки электронной подписи приведены в приложении «В. Генерация и проверка ЭП».

6.3 ПЕРСОНАЛЬНЫЙ СЕРТИФИКАТ

В случае использования персонального сертификата для целей авторизации ВПС на сервере ПЦ, каждый запрос на сервер ПЦ должен происходить с использованием предварительно сгенерированного клиентского сертификата. Генерацию сертификата можно выполнять любой утилитой, например, **openssl.**

Процедура, как правило, следующая:

1. СПП формирует запрос на клиентский сертификат.
2. ПУ генерирует клиентский сертификат, разумеется. Веб-сервер ПУ должен быть настроен на прием клиентского сертификата в качестве средства авторизации.
3. ПУ генерирует две пары ключей. Закрытый ключ клиента и открытый ключ ПУ передаются ВПС, закрытый ключ клиента используется для подписывания запросов на стороне СПП, открытый ключ клиента — для проверки подписи ответов ПУ.

7 НАСТРОЙКА ПРОЦЕССИНГА ДЛЯ РАБОТЫ С ПУ ПО ПРОТОКОЛУ PAYLOGIC

Ключи, логин и пароль или сертификат и пароль от сертификата передаются платежной системе, которая будет присылать запросы, для настройки шлюза. URL запроса в таком случае определяется поставщиком услуги.

Со стороны СПП процессига Paylogic шлюз реализован в ядре системы. Для его запуска необходимо настроить провайдера и шлюз, прописав его в конфигурационном файле шлюзов. Процедура настройки шлюзов изложена в документе по настройке поставщиков и в документе по администрированию ПЦ Paylogic.

Параметры шлюза следующие:

Класс фабрики:

```
ru.softlogic.processing.gates.paylogic
```

Параметры конфигурационного файла для фабрики:

1. **server** — адрес и порт сервера, путь к сертификату (секция keystore опционально) для авторизации. Возможно наличие секции auth-basic с параметрами **username=""**, **password=""** для basic-авторизации;
2. **url** — URL адрес на который следует отправлять запросы на указанном сервере.
3. **balance** — следует ли запрашивать баланс.
4. **signer** — признак, следует ли использовать и проверять ЭП.
5. **public-key** и **private-key** — пути к ключам для подписи запроса и проверки подписи ответа.

Пример конфигурационного файла:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
      <servers>
          <server host="80.0.0.0"
                  port="10443"
                  scheme="https">
                 <keystore path="/home/gates/configs/erc/129.jks"
                           password="123"/>
                 <auth-basic username="task"
                             password="GHtgby&"/>
          </server>
      </servers>
      <gate-config>
          <url value="/server/external"/>
          <balance check="true"/>
          <signer enable="true">
                <public-key
                path="/home/gates/configs/erc/pub.pem"/>
                <private-key
                path="/home/gates/configs/erc/priv.pem"/>
          </signer>
      </gate-config>
</config>
```

8 СПЕЦИФИКАЦИЯ ПРОТОКОЛА

8.1 ЖИЗНЕННЫЙ ЦИКЛ ОПЕРАЦИЙ

Протокол обмена предполагает работу в соответствии с определенным жизненным циклом проведения операций. Жизненный цикл состоит из нескольких этапов, последовательно сменяющих друг друга. Каждому этапу жизненного цикла соответствует определенный запрос, с помощью которого выполняются необходимые действия.

Жизненный цикл операции состоит из двух стадий.

Первая стадия, или стадия ввода данных — предполагает подготовку и проверку допустимости (корректности) введенных данных. На этой стадии выполняются запросы проверки существования абонента — **verify**. Проверки на данном этапе являются опциональными и могут не выполняться, если на стадию проведения будет сформирована операция с неверными или неполными атрибутами, она должна быть поставщиком услуги отклонена (получит в итоге ошибочный статус).

Вторая стадия, или стадия проведения — на этой стадии выполняется регистрация транзакции в системе поставщика услуги, а на стороне СПП — выполняются дальнейшие периодические запросы статуса транзакции, вплоть до получения финального результата проведения платежа (успех или ошибка). На этой стадии выполняются запросы **pay и status .**

Диаграмма действий, выполняемых СПП на этапе проведения платежа — на рисунке 8.1.1.

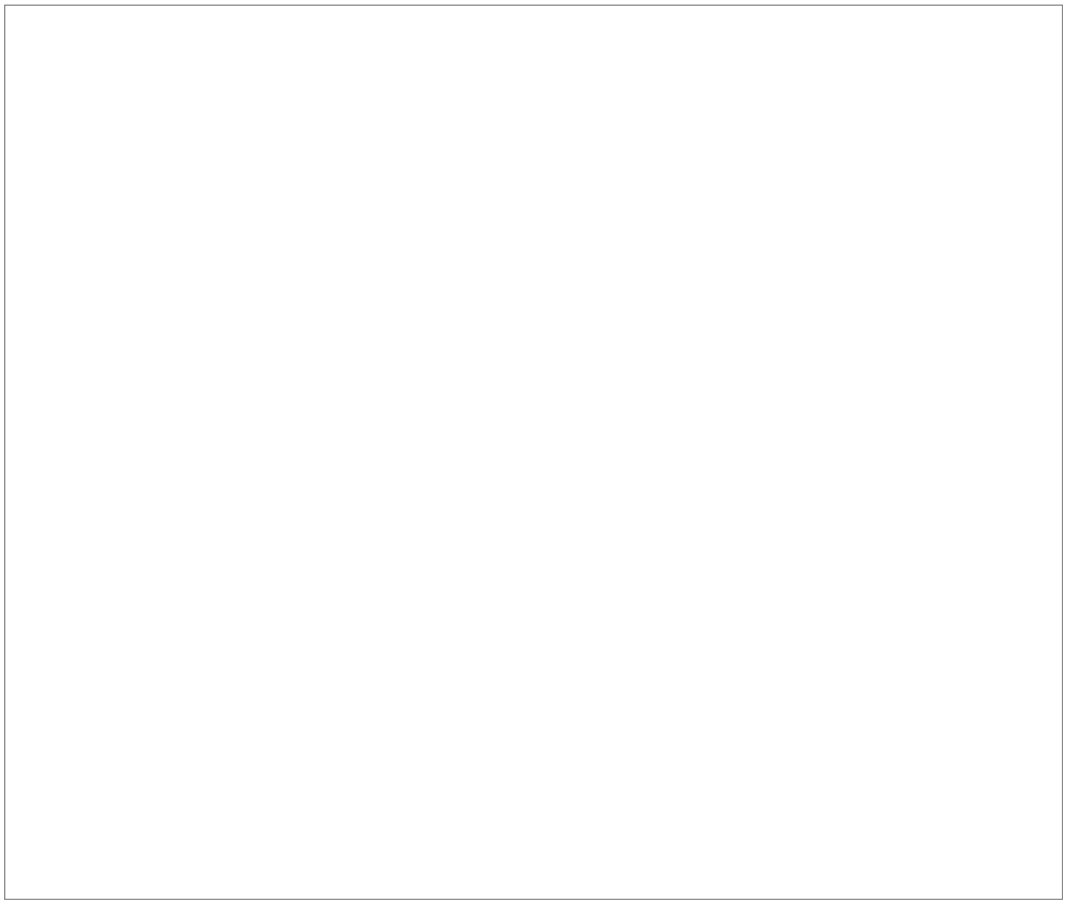

Рисунок 8.1.1 — Схема проведения операции

При реализации протокола следует строго придерживаться приведенных выше схем и строго учитывать обработку финальности. Соответствующие запросы, а также статусы, которые необходимо возвращать ПУ в качестве ответов на запросы проверки, подтверждения и запроса статуса приведены далее в спецификации запросов протокола и таблице статусов операций (Приложение А).

Для СПП финальность или нефинальность определяется не кодом ответа, а признаком Final в ответе. Отсутствие признака означает финальность состояния операции в системе ПУ.

В случае нефинального статуса СПП выдерживает некоторый таймаут, после чего выполняет запрос статуса и анализирует статус и финальность статуса. Цикл повторяется пока ПУ не вернет финальный статус.

В случае финального статуса поведение СПП зависит от того какой статус вернул поставщик. ПУ. Если ПУ вернул код 0 (успех) — платеж считается успешным. Если ПУ вернул код «операция не найдена» (15) — платеж считается ранее не совершенным и СПП проводит его заново, отправляя запрос Payment. Если ПУ вернул любой другой код — платеж считается не проведенным и ставится на стороне СПП во временную ошибку, соответственно со стороны СПП вновь будет выполнен запрос payment через некоторое время.

8.2 ОПИСАНИЕ ФОРМАТОВ ДАННЫХ

Используемые форматы данных приведены в таблице 8.2.1.

Таблица 8.2.1 — Используемые форматы данных

Формат	Описание
intN	Целое N-байтовое число
numeric(N,M)	Дробное, N — количество знаков, M — количество знаков дробной части. Разделитель дробной и целой части — точка
varchar(N)	Строковое, не превышающее длиной N байт. Все строки передаются в кодировке UTF-8
boolean	Логическое, допустимые значения — **true**, **false**
time	Тип данных хранящий дату и время в формате YYYY-MM-DDThh:mm:ss[+|-]HHMI, где: 1. YYYY — год в записанный 4-ми цифрами. 2. MM — месяц с лидирующим нулем. 3. DD — день месяца. 4. T — разделитель. 5. hh — количество часов в 24-ти-часовом формате с лидирующим нулем.

Формат	Описание
	6. mm — количество минут с лидирующим нулем. 7. ss — количество секунд с лидирующим нулем. 8. [+|-]HHMI — часовой пояс.

8.3 СТРУКТУРНЫЕ ЭЛЕМЕНТЫ ПАКЕТА

8.3.1 ОБЩИЕ СВЕДЕНИЯ

Корневым элементом запроса к серверу поставщика является элемент request. Корневым элементом ответа является элемент response.

Запрос может включать следующие элементы:

1. Запрос баланса (<balance>). В одном пакете может быть один такой элемент.
2. Проверка номера абонента (<verify>). В одном пакете может быть один такой элемент.
3. Платеж (<payment>). В одном пакете может быть несколько элементов, но не более 100).
4. Атрибуты платежа (<attribute>). Внутри каждого элемента <payment> может быть произвольное количество атрибутов.
5. Запрос статуса платежа (<status>). В одном пакете может быть несколько элементов, но не более 100.

Ответ поставщика может включать следующие элементы:

1. Баланс лицевого счета агента.
2. Результат операции (<result>). В одном пакете может быть несколько результатов.

8.3.2 КОРНЕВОЙ ЭЛЕМЕНТ — ТЕГ <REQUEST>

Структура элемента запроса приведена в таблице 8.3.2.1.

Таблица 8.3.2.1 — Структура элемента запроса

Название	**Тип**	**Формат**	**Обяз.**	**Описание**
Элемент не имет атрибутов				

Пример:

```
<request>
      ...
</request>
```

Любой запрос должен быть вложен в элемент `<request>`.

8.3.3 БАЛАНС СПП В СИСТЕМЕ ПОСТАВЩИКА — ТЕГ <BALANCE>

В случае запроса баланса СПП должен поместить в пакет пустой элемент `<balance>`. Структура элемента ответа приведена в таблице 8.3.3.1.

Таблица 8.3.3.1 — Структура элемента ответа

Название	**Тип**	**Формат**	**Обяз.**	**Описание**
balance	Атрибут	int8	Нет	Остаток денежных средств на счете ПУ для данной СПП, в копейках.
overdraft	Атрибут	int8	Нет	Кредитный лимит в копейках.

ЗАПРОС АГЕНТА

Запрос должен включать пустой элемент <balance>:

```
<request>
      <balance/>
</request>
```

ОТВЕТ ПЦ

В ответе должен быть элемент <balance> с остатком лицевого счета ВПС:

```
<response>
      <balance balance="1000000" overdraft="0"/>
</response>
```

8.3.4 ПАРАМЕТРЫ ОПЕРАЦИИ — ТЕГ <PAYMENT>

Структура элемента запроса приведена в таблице 8.3.4.1.

Таблица 8.3.4.1 — Структура элемента запроса <payment>

Название	Тип	Формат	Обяз.	Описание
id	Атрибут	int8	Да	Идентификатор операции СПП. СПП должен гарантировать отсутствие у себя в базе повторных операций с одинаковым id. Если поступает операция, которая уже есть в базе в ответ необходимо вернуть текущий статус операции.
sum	Атрибут	int4	Да	Сумма пополнения лицевого счета в копейках.
check	Атрибут	int4	Да	Номер чека, выданного клиенту.
service	Атрибут	int4	Да	Номер услуги.
account	Атрибут	varchar100	Да	Номер абонента в системе поставщика

Название	Тип	Формат	Обяз.	Описание
				услуг.
date	Атрибут	time	Да	Дата поступления платежа в АС ВПС. Используется для сверки.
attribute	Элемент		Нет	Используется для указания дополнительных атрибутов. Может быть несколько. Название атрибутов определяет ПЦ.

Структура элемента <attribute> приведена в таблице 8.3.4.2.

Таблица 8.3.4.2 — Структура элемента <attribute>

Название	Тип	Формат	Обяз.	Описание
name	Атрибут	varchar50	Да	Название атрибута
value	Атрибут	varchar100	Да	Значение атрибута

ЗАПРОС ПРОВЕДЕНИЯ ПЛАТЕЖА

Запрос должен включать один или несколько элементов <payment> возможно с вложенными элементами <attribute>:

```
<request>
      <payment id="14546"
               sum="1000"
               check="17235"
               service="1"
               account="9132345678"
               date="2007-10-12T12:00:00+0300"/>
      <payment id="14547"
               sum="1000"
               check="17235"
               service="2"
               account="12345"
               date="2007-10-12T12:00:00+0300">
          <attribute name="email"
```

```
                        value="info@rol.ru"/>
      </payment>
</request>
```

ОТВЕТ ПЦ

В ответ сервер должен вернуть результат для каждого элемента <payment>:

```
<response>
      <result id="14546"
              code="0"
              final="1"
              trans="123"/>
      <result id="14547"
              code="0"
              final="0"
              trans="312"/>
</response>
```

8.3.5 ПАРАМЕТРЫ ПРОВЕРКИ НОМЕРА АБОНЕНТА — ТЕГ <VERIFY>

Структура элемента запроса приведена в таблице 8.3.5.1.

Таблица 8.3.5.1 — Структура элемента запроса `<verify>`

Название	Тип	Формат	Обяз.	Описание
service	Атрибут	int4	Да	Номер услуги
account	Атрибут	varchar100	Да	Номер абонента в системе поставщика услуг
attribute	Элемент		Нет	Используется для указания дополнительных атрибутов. Может быть несколько. Структура таже как в payment. Описание и состав дополнительных атрибутов для каждого поставщика предоставляется сотрудниками Платежной системы.

ЗАПРОС ПРОВЕРКИ

Запрос проверки номера счета абонента производится в случае онлайн проверки введенных абонентом данных. Запрос должен включать один элемент `<verify>`:

```
<request>
      <verify service="2"
              account="12345">
          <attribute name="email"
                     value="info@rol.ru"/>
      </verify>
</request>
```

ОТВЕТ ПЦ

В ответ сервер должен вернуть результат проверки номера, а также атрибуты от поставщика, в случае их наличия. Состав и возможные значения возвращаемых дополнительных атрибутов определяются особенностями проведения конкретного поставщика, и предоставляются сотрудниками платежной системы.

```
<response>
      <result code="0">
          <attribute name="balance"
                     value="100.00"/>
          <attribute name="fio"
                     value="Пупкин И.В."/>
      </result>
</response>
```

8.3.6 ПАРАМЕТРЫ ПРОВЕРКИ СТАТУСА ПЛАТЕЖА — ТЕГ <STATUS>

Структура элемента запроса приведена в таблице 8.3.6.1.

Таблица 8.3.6.1 — Структура элемента запроса `<status>`

Название	Тип	Формат	Обяз.	Описание
id	Атрибут	int8	Да	Идентификатор операции агента.

В одном запросе может содержаться произвольное количество тегов status. Результат выполнения запроса — тег `<result>`.

ЗАПРОС СТАТУСА

```
<request>
      <status id="123"/>
      <status id="125"/>
</request>
```

ОТВЕТ ПЦ

В ответ сервер должен вернуть результат для каждого элемента <status>:

```
<response>
	<result id="123"
		code="0"
		final="1"
		trans="123"/>
	<result id="125"
		code="15"
		final="0"
		trans="312"/>
</response>
```

8.3.7 ПАРАМЕТРЫ РЕЗУЛЬТАТА — ТЕГ <RESULT>

Структура элемента запроса приведена в таблице 8.3.7.1.

Таблица 8.3.7.1 — Структура элемента запроса <result>

Название	Тип	Формат	Обяз.	Описание
id	Атрибут	int8	Нет	Идентификатор операции СПП. Отсутствует в случае проверки номера абонента.
code	Атрибут	int2	Да	Код ошибки платежа или ошибка проверки реквизитов
final	Атрибут	int2	Нет	Признак финальности статуса запроса (1 — финальный). Отсутствие атрибута означает финальность
trans	Атрибут	int4	Нет	Номер транзакции ПУ

Пример:

```
<result id="14546"
        code="0"
        final="1"
        trans="123456789"/>
```

8.3.8 ОБРАБОТКА НЕПРАВИЛЬНЫХ ПАКЕТОВ

Если по каким-то причинам xml пакет не может быть разобран, в заголовке пакета указаны неверные данные, или произошла внутренняя ошибка на стороне ПУ и т. д. — ПУ должен вернуть результат, корневым элементом которого является элемент error:

```
<error>Package error</error>
```

или:

```
<error>Signature verify error</error>
```

или:

```
<error>Database error</error>
```

9 ИСПОЛЬЗОВАНИЕ ПРОТОКОЛА PAYLOGIC ПОСТАВЩИКАМИ УСЛУГ

9.1 СТРУКТУРНЫЕ ЭЛЕМЕНТЫ ПАКЕТА

Корневым элементом запроса к серверу поставщика является элемент `<request>`. Корневым элементом ответа является элемент `<response>`. Для поставщика услуг элемент `<request>` формируется без каких-либо дополнительных атрибутов, в остальном типы данных и структурные элементы пакета аналогичны спецификации 8.3.2 и 8.3.3, в том числе и количество возможных вложенных элементов пакета запроса и ответа.

Таким образом, запрос может включать следующие элементы:

1. Запрос баланса поставщика (в одном пакете может быть один такой элемент).
2. Проверка номера абонента (в одном пакете может быть один такой элемент).
3. Платеж (в одном пакете может быть несколько элементов).
4. Запрос статуса платежа (в одном пакете может быть несколько элементов).
5. Вложенные атрибуты операции.

Ответ поставщика может включать следующие элементы:

1. Баланс лицевого счета в системе поставщика (в одном пакете может быть один такой элемент).
2. Результат операции (в одном пакете может быть несколько результатов).

Запрос баланса (balance). Необязательный, нужен в случае, когда поставщик ведет баланс платежного агента на своей стороне, и предназначен для получения процессинговым центром информации о текущем балансе, доступном для проведения платежей. Должен быть реализован на стороне поставщика в соответствии со спецификацией 8.3.3. В ответ на запрос баланса поставщиком должен быть сформирован тег ответа с информацией о балансе согласно спецификации на тег `<balance>`.

Запрос проверки номера (verify). Обязательный. Если поставщик не поддерживает на своей стороне такую возможность, ему следует возвращать успешный код ответа. Тег должен быть реализован на стороне поставщика в соответствии со спецификацией 8.3.5. Коды ответов поставщику следует формировать в соответствии с таблицей 9.1.1.

Запрос проведения платежа (payment). Обязательный. Должен быть реализован в соответствии со спецификацией 8.3.4, за исключением признака отложенной операции (delayed) — для поставщика услуги этот признак не передается и не используется.

Обрабатывая запрос Payment следует создавать в своей системе транзакцию и возвращать результат в соответствии с таблицей 9.1.1.

Таблица 9.1.1 — Коды ответов

Код ошибки	Описание
0	Платеж успешно выполнен / Верификация прошла успешно
1	Операция в обработке
2	Нет абонента с таким номером
4	Недостаточно денег на счете агента
5	Невозможно установить соединение с сервером бд
6	Ошибка при работе с БД
10	Неверные параметры
11	Ошибка разбора пакета
15	Платеж не найден
20	Другая ошибка платежного шлюза

Запрос статуса платежа (`<status>`). Обязательный. Необходим для того, чтобы можно было идентифицировать транзакцию и определить ее существование и статус в информационной системе поставщика. Запрос и ответ на него должен быть реализован в соответствии со спецификацией 8.3.6. Статус операции должен соответствовать таблице статусов.

Ответы поставщика услуг на запросы payment и status должны формироваться в соответствии со спецификацией 8.3.7 на тег `<result>` (в теге Result похоже баг), но

при этом поставщиком услуг должны заполняться только два поля ответа: ID (ID операции) и CODE (код ответа), все остальные поля из спецификации 8.3.7 неактуальны для поставщика услуги и в структуре ответа поставщика должны отсутствовать.

ПРИЛОЖЕНИЕ А. РЕЗУЛЬТАТ ДАННЫХ, ВОЗВРАЩАЕМЫХ ПУ

А.1 КОДЫ ОТВЕТОВ

Код ответа	Описание
0	Платеж успешно выполнен / Верификация прошла успешно
1	Операция в обработке
2	Нет абонента с таким номером
4	Недостаточно денег на счете агента
5	Невозможно установить соединение с сервером бд
6	Ошибка при работе с БД
10	Неверные параметры
11	Ошибка разбора пакета
15	Платеж не найден
20	Другая ошибка платежного шлюза

ПРИЛОЖЕНИЕ В. ГЕНЕРАЦИЯ И ПРОВЕРКА ЭП

В.1 ПРИМЕР ГЕНЕРАЦИИ И ПРОВЕРКИ ЭП НА JAVA

```
/**
 * Подписывает строку
 * @param message - Строка для подписи
 * @return подпись в Base64
 */
public String sign(String message) throws SignatureException{
    try {
        Signature sign = Signature.getInstance("SHA1withRSA");
        sign.initSign(privateKey);
        sign.update(message.getBytes("UTF-8"));
        return new String(Base64.encodeBase64(sign.sign()),"UTF-8");
    } catch (Exception ex) {
        throw new SignatureException(ex);
    }
}

/**
 * Проверяет подпись
 * @param message строка для проверки
 * @param signature подись в Base64
 * @return true если подпись верна
 * @throws java.security.SignatureException
 */
public boolean verify(String message, String signature) throws
   SignatureException{
    try {
        Signature sign = Signature.getInstance("SHA1withRSA");
        sign.initVerify(publicKey);
        sign.update(message.getBytes("UTF-8"));
        return sign.verify(Base64.decodeBase64(signature.getBytes("UTF-8")));
    } catch (Exception ex) {
        throw new SignatureException(ex);
    }
}
```

B.2 ПРИМЕР ГЕНЕРАЦИИ И ПРОВЕРКИ ЭП НА PHP

```
   /**
        * @return string
        * @param string $msg
        * @param string $Message
        * @desc подписывает $Message и возвращает электронную подпись. В
          случае ошибки генерирует исключение
        */
    public function Sign($Message){
           $Signature='';
           $KeyId = openssl_get_privatekey($this->PrivateKey,$this-
             >PrivateKeyPass);
           $Res = openssl_sign($Message, $Signature, $KeyId);
           openssl_free_key($KeyId);
           if (!$Res)
               throw new Exception("Sign error!");
           return base64_encode($Signature);
    }
   /**
        * @return none
        * @param string $Message
        * @param string $Digest
        * @desc проверяет соответвие сообщения $Message и подписи $Digest,
             генерирует исключение в случае неудачной проверки
        */
    public function Verify($Message,$Signature)
    {
           $KeyHash = openssl_get_publickey($this->PublicKey);
           $Res=openssl_verify($Message, base64_decode($Signature),
$KeyHash);
           openssl_free_key($KeyHash);
             if ($Res!=1)
                   throw new Exception("Sign verify error");
    }
```