**Задание для кандидата на должность Junior Software Engineer**

**Задание**: Реализовать взаимодействие с Процессинговым центром LeoGaming

**Технологии**: Java EE, Tomcat.

**Вводные данные:**

Адрес для запросов: https://test.lgaming.net/external/extended

Электронно-цифровая подпись: ключи находятся в архиве

Номер точки: point = 327

Номер тестового сервиса: service = 4390

Номер аккаунта: account = любое возможное значение для атрибута

Номер чека: check = любое возможное значение для атрибута

Id операции: любое возможное значение для атрибута

Сумма  0.01 до 15999.00 UAH

В рамках тестовой задачи не обходимо используя ниже описанный протокол взаимодействия реализовать основные операции работы со шлюзом с обработкой ответов от шлюза:

* Verify
* Payment
* Status

**Описание сетевого взаимодействия с шлюзом**

Сетевое взаимодействие должно происходить в рамках протокола https по конкретному адресу методом POST. Запросы и ответы должны быть в формате XML.

Также для гарантии подлинности отправителя и получателя, используется электронно-цифровая подпись запроса и ответа. Для авторизации запроса ЭЦП передается в формате Base64 в HTTP заголовке запроса **PayLogic-Signature**. Для авторизации ответа ЭЦП передается в формате Base64 в HTTP заголовке ответа **PayLogic-Signature**.

Операции необходимо выполнять последовательно:

verify –> payment -> status

1. Все запросы оборачиваются в тег Request:

<request point=”12345”>

…

</request>

Где point обязательный атрибут со значением номера точки приема платежей.

**Тег Verify(проверка введённых данных абонентом):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип | Формат | Обяз. | Описание |
| service | Атрибут | Int4 | Да | Номер услуги |
| account | Атрибут | varchar100 | Да | Номер абонента в системе поставщика услуги |
| attribute | Элемент |  | Нет | Используется для указания дополнительных атрибутов. Может быть несколько. Структура такая же как у Payment. |

Пример:

<verify service=”1” account=”9132345678”>

Пример с дополнительными атрибутами:

<verify service=”2” account=”12345”>

<attribute name=”email” value=[info@rol.ru](mailto:info@rol.ru)/>

</verify>

Структура элемента attribute:

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

**Тег Payment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип | Формат | Обяз. | Описание |
| id | Атрибут | Int8 | Да | Идентификатор операции агента, должен быть уникальным |
| 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 услуг |
| date | Атрибут | Time | Да | Дата поступления платежа, используется для сверки |
| attribute | Элемент |  | Нет | Используется для указание дополнительных атрибутов. Может быть несколько. |

Пример:

<payment id="14546" sum="1000" check="17235"

service="1" account="9132345678" date="2007-10-12T12:00:00+0300"/>

Пример с атрибутами:

<payment id="14546" sum="1000" check="17235"

service="2" account="12345" date="2007-10-12T12:00:00+0300">

<attribute name="email" value="info@rol.ru"/>

</payment>

**Тег 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.

Пример:

<status id=”12345”/>

**Тег Result (ответ шлюза)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **Тип** | **Формат** | **Обяз.** | **Описание** |
| id | Атрибут | int8 | Нет | Идентификатор операции агента. Отсутствует в случае проверки номера абонента. |
| state | Атрибут | int2 | Нет | Статус платежа в системе, возвращается на запрос платежа и статуса |
| substate | Атрибут | int2 | Нет | Субстатус платежа в системе, возвращается на запрос платежа и статуса. |
| code | Атрибут | int2 | Да | Код ошибки платежа или ошибка проверки реквизитов |
| final | Атрибут | int2 | Нет | Признак финальности статуса запроса (1 — фнальный) |
| trans | Атрибут | int4 | Нет | Номер транзакции ПЦ |
| service | Атрибут | Int4 | Нет | Результат проверки реквизитов у поставщика прй вызове advanced запросов. Актуален в случае возврата в атрибуте code значения 0. |
| sum | Атрибут | Int4 | Нет | Сумма платежа в копейках, возвращается при вызове метода сверки платежей |
| commission | Атрибут | Int4 | Нет | Сумма комиссии в копейках, возвращается при вызове метода расчета внешней комиссии |
| sum\_prov | Атрибут | Int4 | Нет | Сумма транзакции в валюте поставщика |
| server\_time | Атрибут | Time | Нет | Время создания операции на сервере ПЦ, может использоваться для сверки с ПЦ |

Пример:

<result id="14546" state="60" substate="0" code="0"

final="1" trans="123456789"/>

**Пример полного цикла проведения операции (запрос ответ):**

***ЗАПРОС ПРОВЕРКИ***

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

<request point="17235">

<verify service="1" account="12345"/>

</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>

***ЗАПРОС ПРОВЕДЕНИЯ ПЛАТЕЖА***

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

<request point="17235">

<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" state="60" substate="0" code="0" final="1" trans="123"/>

<result id="14547" state="40" substate="4" code="0" final="0" trans="312"/>

</response>

***ЗАПРОС СТАТУСА***

Запрос должен включать один или несколько элементов status:

<request point="17235">

<status id="123"/>

<status id="125"/>

</request>

*ОТВЕТ ПЦ*

В ответ сервер должен вернуть результат для каждого элемента status:

<response>

<result id="123" state="60" substate="0" code="0" final="1" trans="123"/>

<result id="125" state="40" substate="4" code="0" final="0" trans="312"/>

</response>

Возможные состояния платежа:

1. 40 — В обработке

2. 60 — Успешно проведен

3. 80 — Ошибка

В случае статуса 80 о причине ошибки мы можем узнать из элемента code.

***Шифрование запросов***

Шифрование будет проводится через header PayLogic-Signature.

Процесс сборки запроса:

- генерируется необходимый XML

- шифруется полученный XML:

          а) полученную xml (полностью все body) ввиде строки  -> достаем все байты

          б) байты кодируем

          в) полученый результат в BASE64 и добавляем в заголовок.

Для шифровки/дешифровки использовать «Примеры генерации и проверки ЕЦП на Java»

- результат шифрования добавляется в header PayLogic-Signature

- отправляется запрос)

***ПРИМЕРЫ ГЕНЕРАЦИИ И ПРОВЕРКИ ЭЦП 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);

}

}