ПРОТОКОЛ ПАКЕТНОГО ОБМЕНА
вер. 4.0

ОПИСАНИЕ
вер. 2.0

МОСКВА
8-495-783-5959

РОССИЯ
8-800-200-0059

ФАКС
8-495-926-4619

WEB
WWW.QIWI.RU

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

Протокол пакетного обмена создан для удобства одновременного проведения большого количества платежей, а также для повышения удобства реализации клиентского ПО за счет переноса части клиентской логики на сервер.

Запросы должны отправляться на один из следующих URL:

- http://service1.osmp.ru/term2/xml.jsp
- https://service1.osmp.ru/term2/xml.jsp
- http://service2.osmp.ru/term2/xml.jsp
- https://service2.osmp.ru/term2/xml.jsp
- http://service3.osmp.ru/term2/xml.jsp
- https://service3.osmp.ru/term2/xml.jsp

Процедура обмена включает следующие этапы:

ШАГ 1. Отправка запроса на проведение платежей

Отправка запросов на сервер осуществляется по HTTP протоколу методом POST.

Пример запроса на проведение платежей:

```
<?xml version="1.0" encoding="utf-8"?>
<request>
	<protocol-version>4.00</protocol-version>
	<request-type>10</request-type>
	<terminal-id>123</terminal-id>
	<extra name="login">login</extra>
	<extra name="password-md5">password-md5</extra>
	<extra name="client-software">Dealer v3.22</extra>
	<extra name="user-agent">1.2.3456.7890</extra>
	<extra name="operating-system">Windows XP (Build 2600)</extra>
	<extra name="serial">ABC12345</extra>
	<auth count="1" to-amount="28.55">
		<payment>
			<transaction-number>123456789</transaction-number>
			<from>
```

```
                        <amount>28.55</amount>
                    </from>
                    <to>
                        <amount>28.55</amount>
                        <service-id>2</service-id>
                        <account-number>(123)456-78-90</account-number>
                    </to>
                    <receipt>
                        <datetime>20060322130025</datetime>
                        <receipt-number>123</receipt-number>
                    </receipt>
                </payment>
        </auth>
</request>
```

Подробное описание запроса приведено в разделе 2.

ШАГ 2. Ответ сервера

Пример ответа сервера:

```
<response requestTimeout=N1>
        <payment status=N2 transaction-number=N3 result-code=N4>
        …
        </payment>
…
</response>
```

Описание ответа приведено в разделе 2.4.

ШАГ 3. Запрос о состоянии платежей

Пример запроса о состоянии платежей:

```
<request>

	…

	<status count=”3”>

		<payment>

			<transaction-number>1</transaction-number>

		</payment>

		<payment>

			<transaction-number>2</transaction-number>

		</payment>

		<payment>

			<transaction-number>3</transaction-number>

		</payment>

	</status>

</request>
```

Описание запроса приведено в разделе 3.

ШАГ 4. Ответ сервера

Пример ответа сервера:

```
<response>

	<payment transaction-number=”N1” status=”N2” result-code=”N3” final-
status=”B1” fatal-error=”B2”>

	…

	</payment>

	…
```

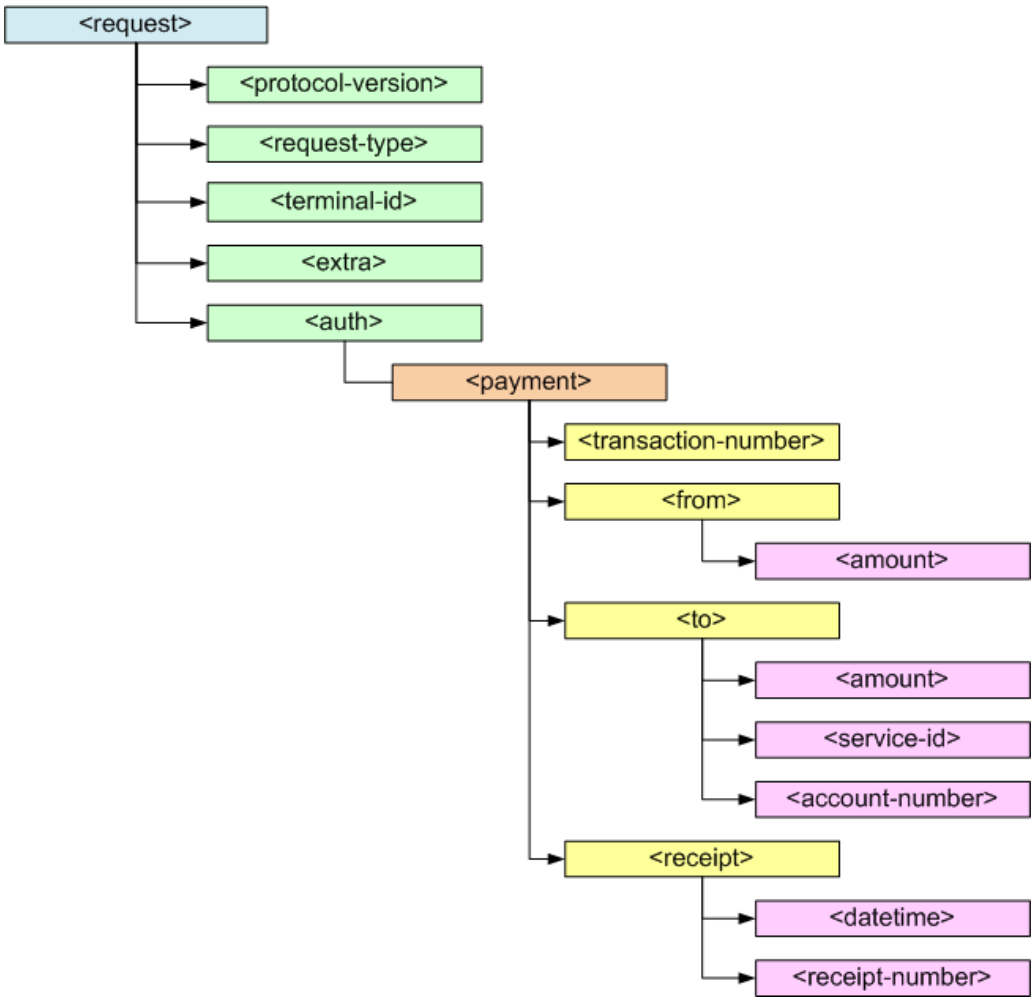
```
</response>
```

Описание ответа приведено в разделе 3.1.

2. ЗАПРОС НА ПРОВЕДЕНИЕ ПЛАТЕЖЕЙ

Структура запроса представлена на Рис. 1.

Рис. 1. Структура запроса



2.1. <request>

Параметры запроса помещаются в элемент `<request>`.

Тег `<request>` включает в себя следующие теги:

- `<protocol-version>` – данный тег содержит номер версии протокола обмена (всегда "4.00");
- `<request-type>` – тип запроса (для пакетного проведения платежей всегда "10");
- `<terminal-id>` – идентификатор терминала, с которого осуществляется платеж;
- `<auth>` – параметры новых платежей на отправку (см. раздел 2.2);
- `<extra>` – тег описывает данные авторизации (Пример 1):

Пример 1. Авторизация

```
<request>
        <protocol-version>4.00</protocol-version>
        <request-type>10</request-type>
        <terminal-id>123</terminal-id>
        <extra name="login">S1</extra>
        <extra name="password-md5">S2</extra>
        <extra name="client-software">Dealer v0</extra>
        …
</request>
```

Где:

- S1 – логин пользователя (кассира)
- S2 – md5 хеш пароля пользователя (кассира). В случае если вы хотите использовать пароль в открытом виде, его нужно посылать в поле "password": <extra name="password">…</extra>.

ПРИМЕЧАНИЕ

Для повышения безопасности вместо пароля в открытом виде или md5 хеша пароля можно использовать специальную подпись <extra name="sign-md5">…</extra>. Подробнее о подписи читайте в Приложении В.

2.2. <auth>

Тег <auth> содержит перечень платежей на отправку. Тег должен содержать атрибуты:

- count – количество тегов <payment> в теге <auth>
- to-amount – сумма всех значений <to><amount>, входящих в состав всех тегов <payment>

ВНИМАНИЕ

В теге <auth> не допускается перечислять больше 50 элементов <payment>

Тег <auth> включает в себя следующие теги:

- <payment> – тег, содержащий данные об одном платеже (может быть несколько тегов):

```
<request>
…
        <auth count=”3” to-amount=”30”>
                <payment>
                …
                </payment>
                <payment>
                …
                </payment>
        </auth>
</request>
```

2.3. <payment>

Тег <payment> описывает конкретные платежи. Тег включает в себя следующие вложенные теги:

- <transaction-number> – номер транзакции на терминале. Пара *номер транзакции* + *номер терминала* должны быть всегда уникальны.
- <from> – информация о сумме, полученной от клиента. Тег содержит дочерний тег <amount>:
  - <amount> – сумма от клиента.
- <to> – информация о платеже: сумма, номер счета, идентификатор провайдера. Тег содержит дочерние теги:
  - <amount> – сумма к зачислению на счет.
  - <service-id> – идентификатор сервиса, на который производится зачисление средств (идентификатор провайдера).
  - <account-number> – идентификатор абонента в информационной системе провайдера (номер сотового телефона или номер лицевого счета).
- <receipt> – информация о выданном чеке. Тег содержит дочерние теги:
  - <datetime> – дата распечатки чека в формате yyyymmddhhnnss.
  - <receipt-number> – номер чека – число от 0 до 999999.
- <extra> – служебные поля, например, комментарий:

```
<extra name=”comment”>это комментарий</extra>
```

Пример 2. Список платежей

```
<payment>
	<transaction-number>N1</transaction-number>
	<from>
		<amount>N2</amount>
	</from>
	<to>
		<amount>N3</amount>
		<service-id>N4</service-id>
		<account-number>N5</account-number>
	</to>
	<receipt>
		<datetime>D1</datetime>
		<receipt-number>N6</receipt-number>
	</receipt>
	<extra name="S1">S2</extra>
</payment>
```

2.4. Ответ сервера на проведение платежей

После отправки запроса на сервер, в ответ приходит пакет с результатами добавления транзакций в базу:

```
<response requestTimeout=N1>
	<payment status=N2 transaction-number=N3 result-code=N4>
	…
	</payment>
…
</response>
```

Где:

- `requestTimeout` – рекомендуемый сервером интервал запроса статуса платежей в секундах. Вы можете использовать это число для отсчета времени (в секундах) перед запросом статуса. К указанному времени все платежи, посланные на обработку, должны быть проведены.

ВНИМАНИЕ

Интервал между запросами статуса желательно устанавливать в соответствии с атрибутом `requestTimeout`

- `status` – статус транзакции после добавления в базу. Нормальным результатом добавления транзакции считается статус 25. Все остальные статусы могут возникать в результате исключительных ситуаций и по вопросам их возникновения необходимо обращаться в службу поддержки.

ПРИМЕЧАНИЕ

В случае если добавление транзакции завершилось со статусом отличным от 25, то необходимо проверить правильность формирования XML запроса и повторить отправку с другим номером транзакции.

- `transaction-number` – номер транзакции, добавленной в базу.
- `result-code` – код обработки операции (см. Приложение А).

3. ЗАПРОС О СОСТОЯНИИ ПЛАТЕЖЕЙ

После добавления транзакции на сервере запускается процедура авторизации и проведения платежа, которая отправляет провайдеру запрос на проверку возможности проведения платежа и в случае положительного ответа провайдера – посылает команду на проведение.

Каждый шаг этой операции сопровождается промежуточными статусами и кодами завершения. После успешного или ошибочного завершения операции, транзакции присваивается определенный статус и код результата завершения.

В любой момент клиентское ПО может обратиться к серверу со следующим запросом для получения текущего статуса транзакции и кода результата завершения:

```
<request>
      <protocol-version>4.00</protocol-version>
       <request-type>10</request-type>
       <terminal-id>123</terminal-id>
       <extra name=”login”>S1</extra>
       <extra name=”password-md5”>S2</extra>
       <extra name=”client-software”>Dealer v0</extra>
       <status count=”3”>
              <payment>
                     <transaction-number>1</transaction-number>
              </payment>
              <payment>
                     <transaction-number>2</transaction-number>
              </payment>
              <payment>
                     <transaction-number>3</transaction-number>
              </payment>
       </status>
</request>
```

Где:

- `<status>` – содержит транзакции, по которым нужно узнать статус. Элемент `<status>` содержит атрибут `count` – количество элементов `<payment>` в теге `<status>`.

ВНИМАНИЕ

В теге status> не допускается перечислять больше 50 элементов <payment>

- <payment> – информация о платеже. Тег содержит дочерний тег:
  - <transaction-number> –номер транзакции, по которой необходимо узнать текущее состояние.

В случае, если на момент обращения к серверу есть и новые транзакции на отправку и транзакции, по которым нужно узнать статус, то возможно (и желательно) комбинирование тегов <auth> и <status> в одном запросе:

```
<request>
…
        <auth>
                <payment>
                        …
                </payment>
                …
        </auth>
        <status>
                <payment>
                        …
                </payment>
                …
        </status>
</request>
```

3.1. Ответ сервера о состоянии платежей

После отправки запроса на сервер, в ответ приходит пакет с результатами обработки запрашиваемых транзакций.

```
<response>
        <payment transaction-number="N1" status="N2" result-code="N3" final-
```

```
status="B1" fatal-error="B2">

      …

      </payment>

      …

</response>
```

Где:

- `transaction-number` – номер транзакции, для которой указан статус;
- `status` – код текущего статуса транзакции (справочник статусов см. в приложении Б);
- `result-code` – код завершения операции (справочник кодов завершения можно получить способом, указанным в приложении А);
- `final-status`:
  - *false* – статус промежуточный и обработка транзакции продолжается, необходим повторный запрос статуса для этой транзакции
  - *true* – статус является окончательным. Обработка транзакции прекращена сервером.
- `fatal-error`:
  - *false* – если значение `final-status` – *true* (сервер прекратил обработку платежа), повторная отправка этого платежа на сервер, но с другим номером транзакции, даст другой результат (возможно платеж пройдет)
  - *true* – обработка платежа завершена в результате фатальной ошибки, повторная отправка платежа с другим номером транзакции приведет к повторению той же ошибки

ПРИЛОЖЕНИЕ А: Запрос кодов завершения

Существует возможность запроса статусов кодов завершения в автоматическом режиме с сервера. Для этого достаточно отправить на сервер запрос следующего вида:

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

<request>

	<protocol-version>4.00</protocol-version>

	<request-type>6</request-type>

	<terminal-id>N1</terminal-id>

	<extra name=”client-software”>Dealer v0</extra>

	<extra name=”login”>S1</extra>

	<extra name=”password-md5”>S2</extra>

</request>
```

Где:

- `N1` – идентификатор терминала
- `S1` – логин пользователя (кассира)
- `S2` – md5 хеш пароля пользователя (кассира). В случае если вы хотите использовать пароль в открытом виде, то его нужно посылать в поле `“password”` (`<extra name=”password”>…</extra>`)

В ответ на запрос сервер вернет справочник кодов следующего вида:

```
<?xml version="1.0" encoding="windows-1251" ?>

<response-codes>

	<response-code id="0">OK</response-code>

	<response-code id="1">Провайдер временно недоступен</response-code>

	…

</response-codes>
```

Где:

- `id` – код завершения
- `<response-code>` – расшифровка кода завершения

ПРИЛОЖЕНИЕ Б: Справочник статусов

Табл. 1. Справочник статусов

Код	Описание
10	Не обработана
20	Отправлен запрос провайдеру
25	Авторизуется
30	Авторизована
48	Проходит финансовый контроль
49	Проходит финансовый контроль
50	Проводится
51	Проведена
58	Перепроводится
59	Принята к оплате
60	Проведена
61	Проведена
120	Не прошла проверки в ОСМП
125	Не смогли отправить провайдеру
130	Отказ от провайдера
148	Не прошел фин. контроль
149	Не прошел фин. контроль
150	Ошибка на терминале
160	Не проведена

ПРИЛОЖЕНИЕ В: Авторизация с помощью подписи

Для обеспечения большей безопасности при пересылке платежных пакетов вы можете использовать авторизацию с помощью специальной подписи (элемент `sign-md5`), которая формируется по следующему алгоритму:

1. Каждый элемент `transaction-number` складывается с MD5 от пароля:

   `transaction-number+password-md5`

1. Осуществляется MD5 хеширование результирующей строки. В результате будет получена строка MD5 последовательности из 32 байт:

   `md5(transaction-number+password-md5))`

2. Все полученные MD5 последовательности (для каждого элемента `transaction-number`) необходимо перевести в битовые массивы, соответствующие шестнадцатеричным кодам, и сложить по модулю 2 (XOR)

`сумма_транзакций=сумма_транзакций XOR md5(transaction-number+password-md5))`

3. Преобразовать полученный в результате всех операций битовый массив в строку, содержащую 32 символа шестнадцатеричных.
4. Полученную строку (длиной 32 символа) необходимо сложить с логином, номером терминала, номером запроса, MD5 от пароля:

   `login+password-md5+terminal-id+request-type+сумма_транзакций`

5. Осуществляется MD5 хеширование результирующей строки

`sign-md5=md5(login+password_md5+terminal-id+request-type+сумма_транзакций)`

Полученная последовательность записывается в элемент `<extra name="sign-md5">`.

Например: `<extra name="sign-md5">81164E9C056CEF02F90E671559D6D4BF</extra>`

Благодаря данному алгоритму авторизации, нет необходимости передавать в запросах элементы `<password>` или `<password-md5>`, что повышает безопасность системы.

Пример 3. Пример формирования подписи на JAVA

```
    /**
    * Function for getting OSMP sign-md5 from Document (Represents source XML)
    * @param doc Документ, сформированный из подписываемой XML
    * @param login Логин пользователя
    * @param passMD5 md5 от пароля пользователя
    * @return Подпись пакета
    * @throws NoSuchAlgorithmException
    */
   private String getSign(Document doc, String login, String  passMD5) throws
```

```
NoSuchAlgorithmException {

        // Получаем id терминала

        String terminal_id = getTextNode(doc, "terminal-id");

        // Получаем тип запроса

        String request_type = getTextNode(doc, "request-type");

        // Получаем суммы транзакций

        String txn_sum =  getSignTxnSum(doc, passMD5);

        // Формируем подпись из строки

        return md5(login + passMD5 + terminal_id + request_type + txn_sum);

    }


    /**

     * Функция получения общей суммы md5 подписей номеров транзакций

     * @param doc Документ, сформированный из подписываемой XML

     * @param passMD5 Документ, сформированный из подписываемой XML

     * @return Строка с md5 подписью номеров

     * @throws NoSuchAlgorithmException

     */

    private String getSignTxnSum (Document doc, String  passMD5) throws
NoSuchAlgorithmException {

        NodeList requestNodeList = doc.getElementsByTagName("request");

        NodeList nl = ((Element)
requestNodeList.item(0)).getElementsByTagName("payment");



        byte[] sum = new byte[16];



        for (int i = 0; i < sum.length; i++)

        {

            sum[i]=0;
```

```
        }


        MessageDigest md = MessageDigest.getInstance("MD5");


        if (nl.getLength()==0)

            return "";


        //Проходим по всем номерам транзакций.

        for (int i = 0; i < nl.getLength(); i++)

        {

            String txn_id = getTextNode(nl.item(i), "transaction-number");
// код ошибки (0 - все OK)

            if (txn_id!=null && txn_id.length()>0)

            {

                //Берем md5 подпись от номера транзакции + пароль пользователя

                byte[] buf = md.digest((txn_id + passMD5).getBytes());

                //Полученный байтовый массив суммируем

                sum = xorArray (sum, buf);

            }

        }


        StringBuffer sb = new StringBuffer();


        // Преобразуем полученный массив в строковое представление в hex

        for (byte aSum : sum) {

            String hex = "00" + Integer.toHexString(aSum);

            sb.append(hex.substring(hex.length() - 2));

        }
```

```
        return sb.toString().toUpperCase();

    }


    //Функция суммирования массивов.

    private byte [] xorArray(byte [] a, byte [] b)

    {

       if(b.length != a.length) throw new IllegalArgumentException("length of
byte [] b must be >= byte [] a");

       byte [] c = new byte[a.length];

       for(int i =0;i< a.length;i++)

       {

          c[i] = (byte)(a[i] ^ b[i]);

       }

       return c;

    }


    //Функция получения строкового представления md5 от строки

    public static String md5(String passw)

    {


        try

        {

            MessageDigest md = MessageDigest.getInstance("MD5");

            byte[] buf = md.digest(passw.getBytes());

            StringBuffer sb = new StringBuffer();


            for (byte aBuf : buf) {

                String hex = "00" + Integer.toHexString(aBuf);

                sb.append(hex.substring(hex.length() - 2));
```

```
            }
            return sb.toString().toUpperCase();

        }
        catch (NoSuchAlgorithmException e)
        {
            return null;
        }
    }


    // Функция получения текстового элемента их xml документа
    public static String getTextNode(Node node, String tagName)
    {
        NodeList nl = null;
        if (node instanceof Element)
            nl = ((Element) node).getElementsByTagName(tagName);
        else if (node instanceof Document)
            nl = ((Document) node).getElementsByTagName(tagName);
        if (nl == null) return null;
        if (nl.getLength() < 1) return null;
        Node n = nl.item(0);
        if (!n.hasChildNodes()) return null;
        n = n.getFirstChild();
        if (n instanceof Text) return n.getNodeValue();
        return null;
    }
```

СПИСОК РИСУНКОВ

СПИСОК ТАБЛИЦ

СПИСОК ПРИМЕРОВ