Додаток 5
до Публічної пропозиції ТОВ «ФК «КОНТРАКТОВИЙ ДІМ» на укладення Договору про надання послуг з приймання та переказу платежів

EasyPay-Provider

V3.1

Зміст

1. Опис

Протокол взаємодії реалізований на основі інтернет-протоколу HTTPS. Дані передаються в форматі xml, використовуючи метод POST. Для захисту каналу передачі даних використовується сертифікат SSL на стороні сервера (провайдера). Всі запити та відповіді підписуються за алгоритмом RSA (SHA1) із довжиною ключа 1024, відповідно сторони (Провайдер, EasyPay) обмінюються сертифікатами у форматі *.PEM.
На даний момент підпис є необов'язковим!

1.1. Загальна структура

Запит

```
<Request>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
        <Operation>
                <Parameter1/>
                 ...
                <ParameterN/>
        </Operation>
</Request>
```

Відповідь

```
<Response>
        <StatusCode>int</StatusCode>
        <StatusDetail>string</StatusDetail>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
        <Parameter1/>
        ...
        <ParameterN/>
</Response>
```

Де

DateTime – час створення запиту (відповіді);
Sign – цифровий підпис у вигляді HEX - рядка (AF4ED0)
Operation – тип операції (список доступних операцій нижче);
StatusCode – код стану (якщо запит виконано успішно, то код 0, якщо помилка, то менше нуля);
StatusDetail – опис результату операції.

2. Типи запитів

- **Check** (перевірка користувача,рахунку);
- **Payment** (створення замовлення);
- **Confirm** (підтвердження замовлення);
- **Cancel** (відміна платежу);

2.1. Перевірка користувача

2.1.1. Запит перевірки акаунта

Запит

```
<Request>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
        <Check>
                <ServiceId>int</ServiceId>
                <Account>string</Account>
                <NfcValidated>True</NfcValidated>

        </Check>
</Request>
```

Відповідь

```
<Response>
        <StatusCode>int</StatusCode>
        <StatusDetail>string</StatusDetail>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
        <AccountInfo>
                <Parameter1>string</Parameter1>
                …
                <ParameterN>string</ParameterN>
        </AccountInfo>
        <FinMon.Recipient>…</FinMon.Recipient>
</Response>
```

ServiceId – номер вашого сервісу в нашій системі;
Account – ідентифікатор користувача (номер телефону, номер договора, …);
AccountInfo – користувальницька інформація, яка відображається на екрані. Значення параметрів - повинні бути на англійській мові (Ці параметри нам повідомляються при створенні сервісу).Через обмеження відображуваної інформації, кількість параметрів не повинно бути більше 4-х;
NfcValidated - для поповнення понад 5000 грн. та валідації клієнта за допомогою NFC у запиті Check передається NfcValidated = True. У відповіді очікується модель ФінМон, за відсутності якого ліміт збільшено не буде (не обязательный, см п.8.4).
FinMon.Recipient – блок фінансового моніторингу (не обов’язковий, див. п.8.4).

2.1.2. Запит отримання номера картки за DPan.

Для отримання номера картки за DPan додатково викликається метод, в якому передається DPan.
Номер картки та DPan шифруються, як у запиті, так і відповіді (див. приклад шифрування 9.5.)
Цей метод викликається до методу Check (п.2.1.1)

```
<Request>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
      <Sign></Sign>
      <GetPanByDpan>
            <ServiceId>int</ServiceId>
            <DPan>OR5Fy8U8jXPVfeR9ybOlr6…</DPan>
      </GetPanByDpan>
</Request>
```

Ответ

```
<Response>
      <StatusCode>int</StatusCode>
      <StatusDetail>string</StatusDetail>
      <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
      <Sign></Sign>
      <Pan>DXzQAzeOa94+BfXw94ocCg8XTSFQ4XRIsf8…</Pan>
</Response>
```

DPan – цифровий номер картки, передається у зашифрованому вигляді (див. п.9.5.)
Pan – номер картки, передається у зашифрованому вигляді (див. п.9.5.)

2.2. Створення замовлення

Запит

```
<Request>
      <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
      <Sign></Sign>
      <Payment>
            <ServiceId>int</ServiceId>
            <OrderId>long</OrderId>
            <Account>string</Account>
            <Amount>decimal</Amount>
      </Payment>
</Request>
```

Відповідь

```
<Response>
      <StatusCode>int</StatusCode>
      <StatusDetail>string</StatusDetail>
      <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
```

```
        <Sign></Sign>
        <PaymentId>string</PaymentId>
</Response>
```

OrderId – це наш унікальний ідентифікатор транзакції (тип long 8-байт);
Amount – сума платежу (у форматі NN,NN, приклад 10.50);
PaymentId – номер платежу у вашій системі.

2.3. Підтвердження замовлення

Запит

```
<Request>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
        <Confirm>
                <ServiceId>int</ServiceId>
                <PaymentId>string</PaymentId>
        </Confirm>
</Request>
```

Відповідь

```
<Response>
        <StatusCode>int</StatusCode>
        <StatusDetail>string</StatusDetail>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
        <OrderDate>yyyy-MM-ddTHH:mm:ss</OrderDate>
        <Info>string</Info>
</Response>
```

OrderDate – дата, яка фокусується у вас, як дата здійснення платежу (списання коштів), вона ж проходить по бухгалтерії и фінансовим документам;
Info – текстова інформація для печаті у чеку (не обов’язковий).

2.4. Відміна замовлення

Запит

```
<Request>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
        <Cancel>
                <ServiceId>int</ServiceId>
                <PaymentId>string</PaymentId>
        </Cancel>
</Request>
```

Відповідь

```
<Response>
```

```
        <StatusCode>int</StatusCode>
        <StatusDetail>string</StatusDetail>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
        <CancelDate>yyyy-MM-ddTHH:mm:ss</CancelDate>
</Response>
```

CancelDate – дата відміни замовлення.

3. Звірка платежів

Постачальнику послуг раз в добу надсилається на email (вказаний при підключенні) реєстр платежів у форматі csv.

Формат файлу:

OrderId;PaymentId;ServiceId;Account;Amount;OrderDate;
11;7891123;223;4589687;45.50;2010-05-02T14:05:30;
12;4139874;497;3257879;5.00;2010-05-02T20:05:30;
14;5478877;544;1121458;15.00;2010-05-02T21:08:30;

4. Цифровий підпис

Для підпису запиту ви повинні використовувати все тіло запиту з пустим тегом <Sign></Sign>:

Запит

```
<Request>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
        <Check>
                <ServiceId>int</ServiceId>
                <Account>string</Account>
        </Check>
</Request>
```

Підпис: RSA1024(SHA1(Запит))

Для перевірки підпису використовується аналогічна процедура:

Відповідь

```
<Response>
        <StatusCode>int</StatusCode>
        <StatusDetail>string</StatusDetail>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
</Response>
```

5. Схеми оплат

5.1. Пряме поповнення

1 Перевірка можливості платежу (перевірка абоненту)

```
<Request>
        <DateTime>2010-09-01T12:00:00</DateTime>
        <Sign>94C8682CABCF1D…</Sign>
        <Check>
                <ServiceId>100</ServiceId>
                <Account>12345678</Account>
        </Check>
</Request>

<Response>
        <StatusCode>0</StatusCode>
        <StatusDetail>OK</StatusDetail>
        <DateTime>2010-09-01T12:00:05</DateTime>
        <Sign>47EDE8324D05CC1…</Sign>
        <AccountInfo>
                <Name>Іванов О.О.</Name>
                <Address>вул. Садова 5, кв. 16</Address>
                <Balance>125.00</Balance>
        </AccountInfo>
</Response>
```

Якщо перевірка виконана успішно переходимо на шаг 2

2 Створення замовлення

```
<Request>
        <DateTime>2010-09-01T12:00:10</DateTime>
        <Sign>2EE9485D8747963E…</Sign>
        <Payment>
                <ServiceId>100</ServiceId>
                <OrderId>11</OrderId>
                <Account>12345678</Account>
                <Amount>25.00</Amount>
        </Payment>
</Request>

<Response>
        <StatusCode>0</StatusCode>
        <StatusDetail>Order Created</StatusDetail>
        <DateTime>2010-09-01T12:00:15</DateTime>
        <Sign>D3479BB615EFF…</Sign>
        <PaymentId>145</PaymentId>
</Response>
```

У разі успіху переходимо на підтвердження платежу.

3 Підтвердження замовлення

```
<Request>
        <DateTime>2010-09-01T12:00:20</DateTime>
        <Sign>615EFFD1D9E02BD…</Sign>
        <Confirm>
                <PaymentId>145</PaymentId>
        </Confirm>
</Request>

<Response>
        <StatusCode>0</StatusCode>
        <StatusDetail>Payment Confirmed</StatusDetail>
        <DateTime>2010-09-01T12:00:25</DateTime>
        <Sign>E4E54DD77C157F…</Sign>
        <OrderDate>2010-09-01T12:00:25</OrderDate>
</Response>
```

При підтвердженні відбувається списання коштів з рахунку партнера. Якщо в час підтвердження відбувся timeout – підтвердження може повторитися, при цьому повинно повертатися те ж, що і в перший раз (OrderDate – при цьому не змінюється).

Якщо додати не обов’язковий тег Info, то текстовий рядок з нього можна надрукувати в чеку

6. Офлайнові платежі

6.1. Формат даних

Інформацію про своїх абонентів (якщо така є) провайдер повинен передавати в наступному форматі:

```
<Clients>
        <Client>
                <Account>string</Account>
                <AccountInfo>
                        <Name>string</Name>
                        <Address>string</ Address >
                        …
                </AccountInfo>
        </Client>
        …
</ Clients>
```

Account – параметр, по якому відбувається пошук в базі абонентів;
AccountInfo – будь-яка інформація про користувача, необхідна, наприклад, для виведення на терміналі

Файл називається як clients-304.xml, де 304 – номер сервісу (послуг), який повідомляється провайдеру при створенні сервісу в системі.

7. Комунальні платежі

Оплата комунальних платежів грунтується на отриманні платіжної інформації, за особовим рахунком або за іншим ідентифікатором. Заповнення лічильників, обчислення пільг, та створення платежу виходячи з введених даних.

Існує два підходи по оплаті рахунків:

- Набір послуг оплачується як один сервіс;
- Кожна послуга сплачується як окремий сервіс;

Відповідно, якщо при перевірці в Products прийшов один тег Product - то сплачується все як одна послуга. Якщо більше одного, то створюється окремо стільки платежів, скільки послуг прийшло з різними ідентифікаторами ServiceId.

7.1. Отримання інформації про платежі

Для отримання інформації про платежі, потрібно передати Account (жек, особовий рахунок , баркод, и т.д.), значення яке буде залежати від конкретної послуги. Якщо ідентифікатор два або більше, то всі вони передаються в операції Check - окремими тегами. Наприклад: <Bill>, <Jek>.

Запит

```
<Request>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign>Sign</Sign>
        <Check>
                <ServiceId>int</ServiceId>
                <Account>string</Account>
        </Check>
</Request>
```

Відповідь

```
<Response>
        <StatusCode>int</StatusCode>
        <StatusDetail>string</StatusDetail>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign>Sign</Sign>
        <Products>
                <Product>
                        <ServiceId>int</ServiceId>
                        <ProductId>string</ProductId>
                        <Address>string</Address>
                        <Account>string</Account>
                        <UserName>string</UserName>
                        <Name lang="ru">string</Name>
                        <Name lang="uk">string</Name>
                        <Date>yyyy-MM-dd</Date>
                        <Amount>decimal</Amount>
                        <PaymentAmount>decimal</ PaymentAmount>
```

```
                    <Amount.Min>decimal</Amount.Min>
                    <Debt>decimal</Debt>
                    <Parameters/>
                    <BankingDetails/>
                    <Meters>
                        <Meter>
                            <Id>string</Id>
                            <Name>string</Name>
                            <Tariff>decimal</Tariff>
                            <Unit>string</Unit>
                            <OldValue>decimal</OldValue>
                            <Privileges>
                                <Privilege>
                                    <Units>decimal</Units>
                                    <Percent>decimal</Percent>
                                </Privilege>
                            </Privileges>
                        </Meter>
                    </Meters>
                </Product>
                …
        </Products>
        <FinMon>…</FinMon>
</Response>
```

де:

Address - жовтим кольором позначені необов’язкові теги;
Products – список можливих послуг;
Product – послуга (містить додаткову інформацію про платежі), якщо надходить декілька послуг, то для кожної послуги буде створюватися новий платіж;
BankingDetails – банківські реквізити (містить інформацію про отримувача платежів), якщо цей параметр не пустий, то він передається в платежі. Цей параметр також може передаватися поза тегом Product, але у тега більший пріоритет. Детально див. пункт 8.1
ServiceId – номер вашого сервісу в нашій системі;
ProductId – динамічний номер послуги (потрібен якщо у межах одного сервісу є декілька вкладених послуг), якщо цей параметр не пусти, то передається в платежі;
Address – поштова адреса платника;
Account – значення параметру Account, яке потрібно вказувати при платежі;
UserName – ім’я платника;
Name – назва послуги, може бути різними мовами;
Date – дата нарахування;
Amount.Min – мінімальна сума для оплати;
Amount.Max – максимальна сума для оплати;
Amount – сума нарахування, без урахування лічильників;
Debt – борг, якщо значення негативне то переплата;
PaymentAmount – сума яку запропонувати сплатити клієнту, використовується тільки на сайті. За замовчуванням використовується Amount
Parameters – додаткова інформація про послугу;

Meters – лічильники;
Meter.Id – ідентифіктор лічильника;
Meter.Name – назва;
Meter.Tariff – тариф;

Meter.Unit – одиниця виміру (кВт., куб…);
Meter.OldValue – останнє значення лічильника, значення може бути пустим;

Privileges – пільги на послугу;
Privilege.Units – кількість одиниць на які діє пільга;
Privilege.Percent – процент, який потрібно сплатити;
FinMon.Recipient – блок фінансового моніторингу (не обов'язковий, див п 8.4), один для всіх послуг.

7.2. Створення платежу

При створенні платежу, передається SessionId , а також параметр PaymentInfo з заповненими значеннями лічильників (якщо такі присутні) або пустий тег <PaymentInfo/>. Додатково передаються ProductId та BankingDetails, якщо вони були вказані при перевірці платежу. Властивість Amount - в лічильниках вказується з урахуванням пільг.

Запит

```
<Request>
	<DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
	<Sign>Sign</Sign>
	<Payment>
		<ServiceId>int</ServiceId>
		<ProductId>string</ProductId>
		<OrderId>long</OrderId>
		<Account>string</Account>
		<Amount>decimal</Amount>
		<PaymentInfo>
			<Meters>
				<Meter>
					<Id>string</Id>
					<OldValue>decimal</OldValue>
					<NewValue>decimal</NewValue>
					<Amount>decimal</Amount>
				</Meter>
				<Meter>
					<Id>string</Id>
					<OldValue>decimal</OldValue>
					<NewValue>decimal</NewValue>
					<Amount>decimal</Amount>
				</Meter>
			</Meters>
		</PaymentInfo>
	</Payment>
</Request>
```

Відповідь

```
<Response>
	<StatusCode>int</StatusCode>
	<StatusDetail>string</StatusDetail>
	<DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
```

```
        <Sign>Sign</Sign>
        <PaymentId>long</PaymentId>
</Response>
```

NewValue – нове значення лічильника

7.3. Підтвердження платежу

Підтвердження платежу відбувається по стандартній схемі

Запит

```
<Request>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
        <Confirm>
                <ServiceId>int</ServiceId>
                <PaymentId>string</PaymentId>
        </Confirm>
</Request>
```

Відповідь

```
<Response>
        <StatusCode>int</StatusCode>
        <StatusDetail>string</StatusDetail>
        <DateTime>yyyy-MM-ddTHH:mm:ss</DateTime>
        <Sign></Sign>
        <OrderDate>yyyy-MM-ddTHH:mm:ss</OrderDate>
        <Info>string</Info>

</Response>
```

8. Додаткові параметри

8.1. Банківські реквізити

Додатковий параметр в операції Check. Повертається провайдером, коли платежі на один сервіс мають різні банківські реквізити.

```
<BankingDetails>
        <Payee>
                <Id>ЄДРПОУ або ІПН отримувача</Id>
                <Name>Назва або ім'я отримувача</Name>
                <Bank>
                        <Name>Назва банку отримувача</Name>
                        <Mfo>МФО отримувача</Mfo>
                        <Account>рахунок отримувача</Account>
                </Bank>
        </Payee>
        <Payer/>
        <Narrative>
                <Name>Призначення платежу у форматі точно з договору</Name>
                <Vat>20</Vat> <!--ПДВ, якщо не береться, то 0-->
        </Narrative>
</BankingDetails>
```

Для того, щоб приймати оплату на декілька отримувачів та розподіляти платіж на декілька отримувачів, на додаток до основних банківських реквізитів на операцію Check потрібно передати у відповіді структуру.

```
<AdditionalPayments>
    <AdditionalPayment>
        <BankingDetails>
          ---------- стандартна структура----------
        </BankingDetails>
        <Rule>
          <Unit>
            ------------------- може бути одне з двох значень "Amount" або "Percent"----------------------
          </Unit>
          <Value>
            --------------------- кількість формату 0.00-----------------
          </Value>
        </Rule>
    </AdditionalPayment>
    <AdditionalPayment>
        ......................
    </AdditionalPayment>
    ........................
</AdditionalPayments>
```

Виконуємо розщеплення платежу на вкладені банківські реквізити згідно з правилом для кожного AdditionalPayment.
Правило-це набір з двох елементів:

- Unit - ознака того, як буде ділитися основна сума. "Amount" - означає, що буде відніматися значення. "Percent" - братися відсоток суми транзакції.
- Value - величина або у відсотках, або у грошах.

8.2. Оригінальний номер сервісу

Додатковий параметр в операції Check, повертається, якщо для даного користувача сервіс поповнення відмінний від того, який вказаний у запиті (можна вказувати лише ті сервіси, які заведені у нас в системі). Це буває необхідно в ситуаціях коли, наприклад, у терміналі сервіс один, а у нас в системі для кожного міста або філії заведений свій (різні договори, банківські реквізити). У цьому випадку покупка пройде з новим ServiceId.

```
<OriginalServiceId>int</OriginalServiceId>
```

Приклад:

```
<Request>
        <DateTime>2010-09-01T12:00:00</DateTime>
        <Sign>94C8682CABCF1D…</Sign>
        <Check>
                <ServiceId>100</ServiceId>
                <Account>12345678</Account>
        </Check>
</Request>

<Response>
        <StatusCode>0</StatusCode>
        <StatusDetail>OK</StatusDetail>
        <DateTime>2010-09-01T12:00:05</DateTime>
        <Sign>47EDE8324D05CC1…</Sign>
        <AccountInfo>
                <Name>Іванов О.О.</Name>
                <Address>вул. Садова 5, кв. 16</Address>
                <Balance>125.00</Balance>
        </AccountInfo>
        <OriginalServiceId>101</OriginalServiceId>
</Response>
```

8.3. Передача суми оплати

Додаткові параметри в операції Check повертаються, якщо для даного користувача фіксована сума до оплати або є нижня межа та верхня, в рамках якої має бути прийнята оплата. Це буває необхідно в ситуаціях, коли у користувачів наприклад різні кредитні заборгованості і він повинен погасити суму не менше зазначеної Y і не має права погасити більше ніж X (або якщо у користувачів різні тарифи, і кожен повинен платити за своїм). У цьому випадку постачальник повинен повернути елементи Amount.Min та Amount.Max.

```
<Amount.Min>decimal</Amount.Min>

<Amount.Max>decimal</Amount.Max>
```

Приклад:

```
<Request>
        <DateTime>2010-09-01T12:00:00</DateTime>
        <Sign>94C8682CABCF1D…</Sign>
        <Check>
                <ServiceId>100</ServiceId>
                <Account>12345678</Account>
        </Check>
</Request>

<Response>
        <StatusCode>0</StatusCode>
        <StatusDetail>OK</StatusDetail>
        <DateTime>2010-09-01T12:00:05</DateTime>
        <Sign>47EDE8324D05CC1…</Sign>
        <AccountInfo>
                <Name>Іванов О.О.</Name>
                <Address>вул. Садова 5, кв. 16</Address>
                <Balance>125.00</Balance>
        </AccountInfo>
        <Amount.Min>125.00</Amount.Min>
        <Amount.Max>125.00</Amount.Max>
</Response>
```

*Якщо значення Amount.Min та Amount.Max рівні клієнт зможе сплатити лише зазначену суму у значеннях цих параметрів, якщо немає тега в діапазоні від Amount.Min до Amount.Max.

8.4. Блок фінансового моніторингу

Відповідно до ст. 14 Закон України Про запобігання та протидію легалізації (відмиванню) доходів, отриманих злочинним шляхом, фінансуванню тероризму та фінансуванню поширення зброї масового знищення (проект 2179) усі перекази мають супроводжуватись інформацією про платника (ініціатора переказу).

Вимоги статті 14 не поширюються на операції з переказу коштів на суму менше ніж 5000 гривень.

Переказ у сумі від 5000 до 30000 грн (29 999,99 грн), при наданні потрібних параметрів поширюється на платежі з метою оплати вартості товарів, робіт, послуг, погашення заборгованості за кредитом для зарахування на рахунок одержувача.

Виняток:

Переказ коштів з метою сплати податків, зборів, платежів, зборів на обов'язкове державне пенсійне та соціальне страхування, штрафних санкцій та пені за порушення законодавства до державного та місцевих бюджетів, Пенсійного фонду на рахунки органів державної влади, органів місцевого самоврядування або переказу коштів за житлово -комунальні послуги (до 29 999,99 грн. без проведення дод. ідентифікації)

Якщо картка клієнта була провалідована за допомогою NFC на терміналі, то передається NfcValidated = True. У відповіді очікується модель FinMon.Recipient.

Якщо постачальник хоче, щоб по сервісу клієнт мав можливість здійснювати платежі сумарно більш ніж 5000 грн на день (до 29 999,99 грн.), то постачальник повинен нам повернути на операції Check наступний блок:

<FinMon.Recipient>

<FullName>Іванов Іван Іванович</FullName> - поле обов'язкове, далі один із параметрів на вибір, але обов'язково один має бути заповнений:

<Id>1234567891</Id> - реєстраційний номер облікової картки платника податків

<Residence>Бровари, Київска, 243, кв.14</Residence> - місце проживання

<Passport>АН 123456</Passport> - номер (та за наявності - серію) паспорта громадянина України

<BirthDate>10.01.1993, Житомир</BirthDate> - дата і місце народження

</FinMon.Recipient>

Приклад Check:

```
<Request>
        <DateTime>2020-04-22T09:07:18</DateTime>
        <Sign>…</Sign>
        <Check>
                <ServiceId>int</ServiceId>
                <Account>string</Account>
        </Check>
</Request>

<Response>
        <StatusCode>0</StatusCode>
        <StatusDetail>OK</StatusDetail>
        <DateTime>2020-04-22T09:07:18</DateTime>
        <Sign>…</Sign>
        <Check>
                <ServiceId>int</ServiceId>
                <Account>string</Account>
        </Check>
</Request>

<Response>
        <StatusCode>0</StatusCode>
        <StatusDetail>OK</StatusDetail>
        <DateTime>2020-04-22T09:07:18</DateTime>
        <Sign>…</Sign>
        <FinMon.Recipient>
                <FullName>Іванов Іван Іванович</FullName>
                <Id>1234567891</Id>
                <Residence></Residence>
```

```
                    <Passport>АН 123456</Passport>
                    <BirthDate></BirthDate>
                </FinMon.Recipient>
</Response>
```

8.5. Розділення однієї послуги на декілька транзакцій (сплітування)

Всі раніше описані схеми мають на увазі, що на 1 (одну) послугу в нашій системі створюється тільки 1 (одна) транзакція як для схеми прямого поповнення, так і для схеми з продуктами (1 транзакція на кожен продукт)

Щоб послугу, яку оплачує замовник, розділити на декілька окремих транзакцій, потрібно використовувати додатковий тег <SplitItem>. Цей механізм працює тільки зі схемою з Продуктами. Тому, якщо треба розділити послугу, її треба надати, як продукт, і додати тег <SplitItem>, який ділить цей продукт на окремі транзакції. Може бути багато продуктів і відповідних до кожного сплітів. Якщо продукт не треба розділяти, то <SplitItem> для нього не додається і він буде оплачений за звичайною схемою.

Приклад структури:

```
<Response>
        <StatusCode>0</StatusCode>
        <StatusDetail>string</StatusDetail>
        <DateTime>2022-11-12T11:55:33</DateTime>
        <Sign></Sign>
        <AccountInfo>
                </Parameter1>string</Parameter1>

         …

                <ParameterN>string</ParameterN>

          <AccountInfo>

          <Products>

                  <Product>

                          </ProductId>ParentProduct1</ProductId>
                          <ServiceId>12345</ServiceId>
                            <Account>AccountToDisplayOnFront</Account>
                            <UserName>FOP Ivanov</UserName>
                            <BankingDetails>    <!-- Реквізити з 1 спліта -->

                                    <Payee>
                                        <Id>12345678</Id>
                                          <Name>ТЕСТ</Name>
                                          <Bank>
                                                <Name>БАНК</Name>
                                                <Account>UA193052990000026005045001234</Account>
```

```
                    </Bank>
                </Payee>

                <Narrative>
                    <Name>за послуги</Name>
                </Narrative>

            </BankingDetails>
            <Name lang="uk">за послуги</Name>
            <Amount.Min>300.00</Amount.Min>
            <Amount.Max>300.00</Amount.Max>
            <Amount>300.00</Amount>
            <Date>2021-12-22</Date>

        </Product>

        <Product>

            <ProductId>ParentProduct2</ProductId>
            <ServiceId>12345</ServiceId>
            <Account>AccountToDisplayOnFront</Account>
            <UserName>FOP Ivanov</UserName>
            <BankingDetails>    <!-- Реквізити з 1 спліта -->

                <Payee>

                    <Id>12345678</Id>
                    <Name>ТЕСТ</Name>
                    <Bank>

                        <Name>БАНК</Name>
                        <Account>UA193052990000026005045001234</Account>

                    </Bank>

                </Payee>

                <Narrative>

                    <Name>за послуги</Name>

                </Narrative>

            </BankingDetails>
            <Name lang="uk">за послуги</Name>
            <Amount.Min>300.00</Amount.Min>
            <Amount.Max>300.00</Amount.Max>
            <Amount>300.00</Amount>
            <Date>2021-12-22</Date>

        </Product>

    </Products>
    <ServiceCache>

        <SplitItem>

            <ProductId>ParentProduct1</ProductId> <!-- ProductId базового зі списку
            продуктів →
            <Rule>Amount</Rule> <!-- правило для створення транзакцій по базовому
            продукту по сумі: сума сплітів = сумі базового продукту, суми статичні  →
```

```
        <Split>

            <SplitProductId>ChildProduct11</SplitProductId> <!-- ProductId спліта,
            на оплаті він буде переданий замість базового продукта -->

        <Account>AccountForTransaction</Account>
        <Value>200.00</Value>
        <BankingDetails>  <!-- реквізити для зарахування -->

            <Payee>

                    <Id>12345678</Id>
                    <Name>ТЕСТ</Name>
                    <Bank>
                            <Name>БАНК</Name>

                    <Account>UA193052990000026005045001234</Account>

                    </Bank>

            </Payee>

            <Narrative>

                    <Name>за послуги</Name>

            </Narrative>

</BankingDetails>

        </Split>

        <Split>

            <SplitProductId>ChildProduct12</SplitProductId>
                <Account>AccountForTransaction</Account>
                <Value>100.00</Value>
                <BankingDetails>  <!-- реквізити для зарахування -->

              <Payee>

                    <Id>12345678</Id>
                    <Name>ТЕСТ</Name>
                    <Bank>

                        <Name>БАНК</Name>

                        <Account>UA193052990000026005045001006</Account
                        >

                    </Bank>

              </Payee>

              <Narrative>

                    <Name>за послуги</Name>

              </Narrative>

            </BankingDetails>
```

```
        </Split>
</SplitItem>


<SplitItem>
        <ProductId>ParentProduct2</ProductId>
        <Rule>Amount</Rule>  <!-- правило для створення транзакцій по
        базовому продукту по сумі: сума спітів = сумі базового продукту, суми
        статичні  →
    <Split>
        <SplitProductId>ChildProduct21</SplitProductId>
        <Account>AccountForTransaction</Account>
        <Value>200.00</Value>
        <BankingDetails>  <!-- реквізити для зарахування →
            <Payee>
                <Id>12345678</Id>
                <Name>ТЕСТ</Name>
                <Bank>
                    <Name>БАНК</Name>
                    <Account>UA193052990000026005045001234</Account>
                </Bank>
            </Payee>
            <Narrative>
                <Name>за послуги</Name>
            </Narrative>
        </BankingDetails>
    </Split>
    <Split>
        <SplitProductId>ChildProduct22</SplitProductId>
        <Account>AccountForTransaction</Account>
        <Value>100.00</Value>
        <BankingDetails>  <!-- реквізити для зарахування →
            <Payee>
                <Id>12345678</Id>
                <Name>ТЕСТ</Name>
                <Bank>
                    <Name>БАНК</Name>
                <Account>UA193052990000026005045001009</Account>
                </Bank>
            </Payee>
```

```
                    <Narrative>
                        <Name>за послуги</Name>
                    </Narrative>
                </BankingDetails>
            </Split>
            <Split>
        <ServiceCache>
</Response>
```

Спліти працюють тільки в рамках одного ServiceId і поки реалізовано розділення тільки по статичним сумам, тобто базовий продукт і його спліти мають фіксовані суми.

Для структури є обов'язкові перевірки:

1. Для всіх сплітів має бути базовий (Parent) продукт.
2. Сума спітів по одному базовому продукту має дорівнювати сумі його сумі (суми статичні).
3. Наявність реквізитів у сплітах і у базовому продукті - обов'язково (мають бути валідні).

На операції Payment при оплаті сплітованого продукту, будуть відправлятися окремі оплати з унікальним OrderId і відповідним SplitProductId для кожного спліта:

```
<Payment>
    <ServiceId>int</ServiceId>
    <ProductId>ChildProduct11</ProductId>
    <OrderId>long</OrderId>
    <Account>AccountForTransaction</Account>
    <Amount>200.00</Amount>
    <PaymentInfo>…</PaymentInfo>
</Payment>
```

9. Доповнення А

9.1. Створення сертифікату

Для створення сертифікату підпису, можна скористатися безкоштовною утилітою openssl.

```
openssl genrsa -out provider.ppk 1024
openssl req -new -key provider.ppk -out provider.req
openssl x509 -req -days 730 -in provider.req -signkey provider.ppk -out provider.cer
```

При створенні сертифікату підпису використовується CN=ProviderSign-ProviderName, де ProviderName – назва провайдера латинськими буквами (наприклад, CN=ProviderSign-TopNet)

9.2. Підпис даних

Приклад підпису на C#:

```
string path = "Provider-Test.pfx";
X509Certificate2 cert = new X509Certificate2(path,"test");
RSACryptoServiceProvider rsa = (RSACryptoServiceProvider)cert.PrivateKey;

byte[] signature = rsa.SignData(Encoding.UTF8.GetBytes(data), new SHA1CryptoServiceProvider());
string result = Utilities.BytesToHex(signature);
```

9.3.Тестовий хост

Перед бойовим запуском Ви повинні пройти тестування на сторінці https://provider.easypay.ua

9.4. Авторизаційні дані на стороні провайдеру

Провайдер повинен заздалегідь проінформувати про те, якого типу авторизація передбачена на сервері (сертифікат, логін та пароль і т.д.)

9.5. Шифрування/дешифрування облікового запису

Шифрування використовується для шифрування/дешифрування облікового запису, номера картки та DPan на методах Check, GetPanByDpan та Payment.

EasyPay на своїй стороні шифрує публічним сертифікатом постачальника, а постачальник дешифрує дані своїм приватним сертифікатом (треба використовувати RSA/PKCS1Padding).

У відповіді постачальник на своїй стороні шифрує дані публічним сертифікатом EasyPay.

Приклад шифрування облікового запису:

```
public string EncryptString(X509Certificate2 certificate, string data)
  {
        var provider = (RSACryptoServiceProvider)certificate.PublicKey.Key;
        var dataBytes = Encoding.ASCII.GetBytes(data);
        var encryptedBytes = provider.Encrypt(dataBytes, false);
        return Convert.ToBase64String(encryptedBytes);
  }
```

Приклад дешифрування облікового запису:

```
public string DecryptString(X509Certificate2 certificate, string data)
  {
        var provider = (RSACryptoServiceProvider)certificate.PrivateKey;
        var dataBytes = Convert.FromBase64String(data);
        var encryptedBytes = provider.Decrypt(dataBytes, false);
        return Encoding.ASCII.GetString(encryptedBytes);
```