PayRun API

описание подключения к сервисам PayRun

версия 1.2.3

Оглавление

Цель

Данный документ описывает механизм подключения к сервисам XPAY с целью получения сервисов оператора:

1. проведения платежей
2. получение статуса платежа
3. открытие кошелька пользователя
4. получение баланса по кошельку
5. получение выписки по кошельку
6. запрос эмиссии
7. запрос погашения электронных денег
8. получения токена карты
9. регистрации операции списания с карты по токену

Общее описание мер безопасности

Взаимодействие с информационной системой оператора требует соблюдения следующих мер безопасности:

1. сетевые подключения выполняется в рамках пула разрешенных IP адресов партнера.
2. обмен происходит в рамках защищенного протоколом SSL соединения.
3. пакет данных подлежит обязательному шифрованию с использованием персонального публичного ключ партнера.
4. идентификация партнера выполняется следующим образом:
    a. уникальный токен;
    b. подпись пакета данных приватным ключем партнера и верификация подписи на стороне оператора с помощью публичного ключа партнера.
    c. криптование пакета публичным ключом, который генерируется на стороне оператора и передается партнеру.

Для этого:

1. Оператор передает партнеру его токен и свой публичный ключ;
2. Партнер предоставляет оператору свой публичный ключ;

*требование к ключам шифрования: RSA ключ длиной 2048 бит, упакован PKIX алгоритмом, в pem формате.

Пример команд для генерации ключа и формирования публичной части:

openssl genrsa -out keypair.pem 2048

openssl rsa -in keypair.pem -pubout -out publickey.crt

Интеграция Google Pay™

Интеграция

Поддерживается 2 типа интеграций: DIRECT и PAYMENT_GATEWAY. С общей инструкцией по подключению Google Pay™ можно ознакомиться здесь.

DIRECT

Для получения merchantID необходимо следовать инструкциям описанным здесь.

PAYMENT_GATEWAY

Для GATEWAY интеграций необходимо в параметре gateway использовать значение payrun («gateway»: «payrun»). gatewayMerchantid мы предоставим в процессе технической интеграции.

Ресурсы по Android интеграции: Google Pay Android developer documentation, Google Pay Android integration checklist и Google Pay Android brand guidelines.

Ресурсы по web интеграции: Google Pay Web developer documentation, Google Pay Web integration checklist и Google Pay Web Brand Guidelines.

Поддерживаемые страны:
- Ukraine

Поддерживаемые платежные системы:
- Visa
- Mastercard

Проведение платежей

В качестве paymentMethod поддерживается только CARD. В качестве способа проведения платежа поддерживаются PAN_ONLY и CRYPTOGRAM_3DS. По умолчанию все PAN_ONLY транзакции будут автоматически направлены на 3d-secure банка эмитента, иная логика согласовывается по запросу. Для проведения платежа необходимо получить от Google™ платежный токен методом PaymentDataRequest Затем полученный токен необходимо в полном виде передать в блоке pay_params в методе Карта-Кошелёк, структура TransferC2W.

Параметр BillingAddressParameters является опциональным, на проведение платежей не влияет.

Шифрование

Поддерживаются ECv2 версия протокола шифрования Google™ –, для успешной обработки платежа необходимо передать корректный платежный токен с указанной в ней версией протокола.

URI

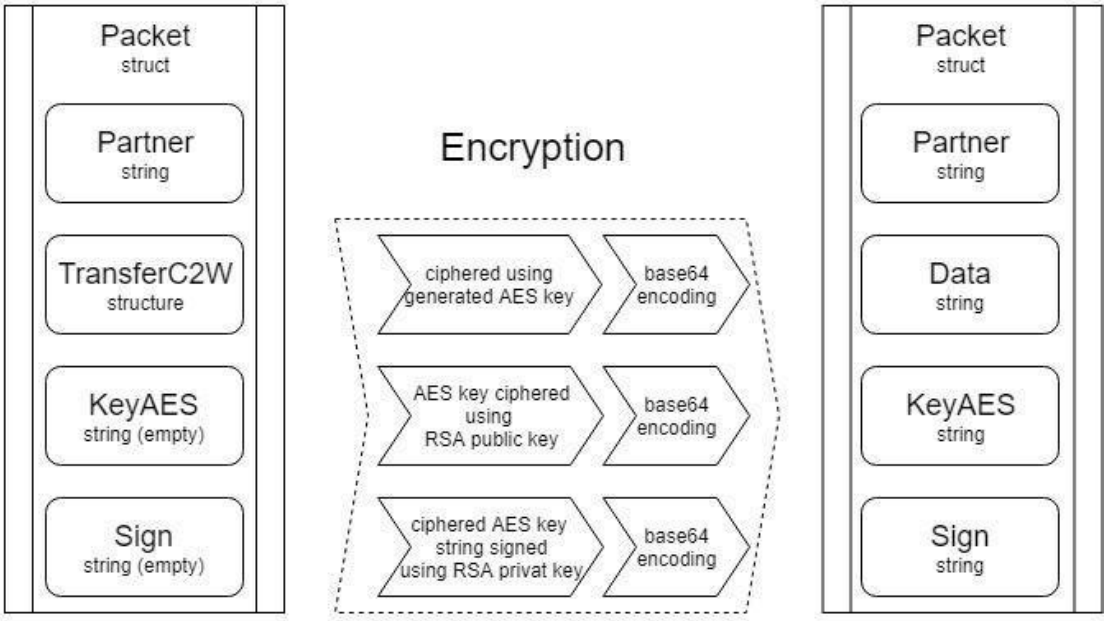
Сервисы расположены по URI
https://papi.payrun.online/payrun

Сервисы тестовой среды расположены по URI
https://stage-papi.payrun.dev/payrun

Запрос

Структура запроса

Схема формирования запроса:



Пакет запроса представляет собой JSON структуру состоящую из 4-х полей:

1. "**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":999,
   "Locale":""
}
```

Поле	**Тип**	**Описание**
PartnerToken	Строка	Токен партнера

OperationType	Целое	Код типа операции
Locale	Строка	Локаль сообщений. Допустимые значения: “en” - английская “uk” - украинская (значение по умолчанию) “ru” - русская

2. “**Data**" - зашифрованные по алгоритму AES данные операции (строка).

   Для формирования данного поля необходимо:
   - генерируется ключ длиной 16 байт.
   - данные структуры шифруются по алгоритму AES-128 в режиме CBC:
     - размер блока 16 байт
     - выравнивание текста по размеру блока
     - к незашифрованному выравненному тексту, в первые 16 байт, добавляется случайно сгенерированный вектор инициализации.

       полученный в результате шифрования бинарный массив стандартно преобразовать в строку в кодировке base64.
3. "**KeyAES**" - зашифрованный по алгоритму RSA, ключ AES.

   Для формирования данного поля необходимо:

   - сгенерированный ключ, которым зашифрованы данные операции, зашифровать публичным ключем PayRun:
     - алгоритм PKCS1 v1.5
   - полученный в результате шифрования бинарный массив стандартно преобразовать в строку в кодировке base64.

4. "**Sign**" - подпись по алгоритму RSA

   Для формирования данного поля необходимо:
   - зашифрованный ключ (см. п.3), до преобразования по base64, подписывается приватным ключем:
     - хеш SHA256
     - алгоритм PKCS1 v1.5
   - полученный в результате подписи бинарный массив стандартно преобразовать в строку в кодировке base64.

Результатом выполнения запроса является ответ, структура которого описана в разделе “Структура ответа”.

Правила отправки запросов

Получение http кода с признаком фатальности = “нет” подразумевает продолжение выполнения операции в системе оператора или партнера. Для запроса текущего статуса обработки операции необходимо повторять запрос с направленными ранее параметрами до получения http кода с признаком фатальности “да”, см. раздел “Коды ответов”. В ответе будет меняться статус проведения транзакции.

Направление повторного запроса оператору не должно быть чаще чем 1 раз в 5 секунд.

Ответ

Структура ответа

В ответ на запрос операции, сервис отвечает JSON структурой, состоящей из 3-х полей:

```
{
   "Code":,
   "Message":"",
   "Data":{},
   "KeyAES":"",
   "Sign":""
}
```

Поле	Тип	Описание
Code	Целое	http статус выполнения запроса. Доступные статусы указаны в разделе “Коды ответов”
Message	Строка	Текстовое сообщение, с описанием статуса запроса
Data	Структура	Вспомогательные данные операции (идентификатор операции, URL подтверждения операции, выписка баланса, выписка операций...)
KeyAES	Строка	Зашифрованный по алгоритму RSA, ключ AES которым зашифрованы данные операции (полученный в результате шифрования бинарный массив преобразуетс в строку в кодировке base64). Для ответов с не шифрованными данными - пустая строка
Sign	Строка	Подписанная по алгоритму RSA строка “KeyAES” (полученный в результате подписания бинарный массив

		преобразуется в строку в кодировке base64). Для ответо с не шифрованными данными - пустая строка.

Массив строк Data для запросов с http кодом ответа = 102 содержит строки:
"OperationID: код операции",
"OperationStatus: статус выполнения операции"

код операции - уникальный идентификатор операции в системе оператора.
статус выполнения операции - код статуса операции, приведен в разделе “Статус обработки операции”.

Примеры ответов:

- получен неправильный токен:
  {"Code":401,"Message":"wrong token","Data":null,"KeyAES":"","Sign":""}

- запрошена недопустимая операция:
  {"Code":403,"Message":"operation not permitted","Data":null,"KeyAES":"","Sign":""}

- Операция успешно проведена:
  {"Code":200,"Message":"operation ok","Data":{"OperationID":11,"OperationStatus":
  10},"KeyAES":"","Sign":""}

- Зашифрованный ответ баланса и выписки по счету имеют вид:
  {"Code":200,"Message":"done","Data":{"encodedData":"WPr8Askp1cir7LyF43W8iXswDk
  gQ3IUx9LUSM3n8w+0Uh0mh5eLW8eAZeEuFyYxECPJKwQQIEU8JaITcdw2FbAWWE
  ptokVWYzX8U3mGE3UZBk+FN8gl7o8wiF6ZUl/Wqo4c1Ukqfk1Iu7atpigKsDH5GZ7uaQr
  Sj2syOSpJhBccQ3sUqiMxNNHxP/cUnyOBQRKbCVsE="},"KeyAES":"UE+AGP1kcDSG
  skefiJM1u41SHDin/8NDbiXex2EdizbWRF5L9IPtfVD9MSbx6ZfZmV7LQtrwqo4Ka7BhRi
  E8kGggbiWWmOkdliThcFCR/FUXoc/X5JBKzYVxYXrHTgPZn93Zi17eL7P1nwBaC2xsn
  ekkbOe3VC9XwTGh6LVP0DAsZ5HmXC1CSAXpbuhNEDFT5oOiE/vCJczUU/+aXnSRJ
  qM9YexAgoKSGLRNtI3HHWVtdYBJFqSpOp61wZ5+CumvTvBeTfk0j0zXZyNduJxXKUiv
  06o9OZRIGJ5Hkmopeq6QqhgokRBTvXfHGoja7sMJ+g/v1grwktDdsbrv9xuGmaA==","Si
  gn":"UQgMeDmD6FGvA4kmtva/0rjf9p9KYf8ZM4m/kU6PpyEPoTG9rdlCu5xe4Yaba8y/6
  u2NyM6HNyyZ3luj80I1ati83jq4ZQo8F6D9hiELrFY/U2NMonTvSTGXU4GBi1Ca8OSc5V
  H3PY7RBxfX9O1iivPFIRuhcdi5ZV5cPvyPLRKFi2/ArZ4omjTWK3FLcX0hA47YI5xRjP1u
  5YzOM/HnGahLwUc3vux8yMKejKy43IRPDrWlIa2uuX6npr3Ao+/Mcc8OrKRO6RO2BQii
  FRGaiPQttBdZ9p8qLCf2EpjiVYGJMtd/F31EYjqxDQ4X/+ALxuc2y4F3zjvWsh8DZA/Sjg=
  ="}

Коды ответов

Код отве	Сообщение	Описание	Фатальность
102	Processing	Платеж в обработке	нет
200	OK	Обработка операции завершена успешно	да
400	Bad Request	Ошибка парсинга запроса. Ошибка структуры	да
401	Unauthorized	Токен не соответствует ключу	да
403	Forbidden	Выполнение операции запрещено	да
404	Not Found	Метод не найден	да
500	Internal Server Error	Произошла непредвиденная ошибка выполнения протокола	да
503	Service Unavailable	"Сервис недоступен". Сервер не готов обрабатывать запрос.	да

Статус обработки операции

Код	Статус
1	Принят для обработки
2	Проверка карты на 3DS
3	Ожидание верификации OTP
4	Операция в очереди
5	Операция в обработке
7	Ожидание подтверждения
10	Операция проведена
20	Ошибка верификации OTP
21	Ошибка проведения операции
22	Операция отменена

Коды причин отклонения операций

1	Платеж отклонен по временным техническим причинам. Пожалуйста, повторите оплат позже. Техническая проблема
2	Абонент не найден
3	Платеж отклонен. На данной карте недостаточно средств для совершения транзакции Недостаточно средств на карте
6	Платеж отклонен. Проверьте правильность введённых реквизитов карты или укажите

	номер другой карты Отказ банка эмитента
9	Платеж отклонен. Проверьте правильность введенных реквизитов карты. Неправильн введен CVV
11	Платеж отклонен. Попробуйте оплатить другой картой или повторите оплату позже. Антифрод банка эквайера
12	Платеж отклонен. Срок действия Вашей карты истек. Обратитесь в банк.
13	Платеж отклонен. Превышен суточный лимит количества операций по карте. Пожалуйста, обратитесь в Ваш банк.
14	Платеж отклонен. Карта не поддерживает дополнительный уровень безопасности 3DSecure. Просьба обратиться в Ваш банк для подключения функции.
18	Платеж отклонен. Проверьте правильность введённых реквизитов карты или укажите номер другой карты или обратитесь в банк, выдавший Вашу карту. Карта заблокирова эмитентом
19	Платеж отклонен. Вы отказались от оплаты платежа
22	Лимит оплат в интернете по Вашей карте превышен. Пожалуйста, обратитесь в Ваш банк.
27	Платеж отклонен. Страна карты отличается от страны сетевого адреса компьютерной сети (IP).
42	Истекло время отведенное на оплату
61	Операция по Вашей карте в Интернете была отклонена из-за недостаточного интернет-лимита по сумме операций. Рекомендуем Вам увеличить интернет-лимит в Приват24 (меню «Мои счета» > «Управление картой/счетом» > «Оплата в Интернете») повторить операцию немного позже.
63	Проверка 3D-Secure не выполнена. Просьба обратиться в Ваш банк для уточнения информации.
64	Неправильный проверочный код
88	Другая ошибка
99	Превышен лимит операции (14 000 грн)

Обработка запроса (CallbackURL)

В процессе выполнения платежа производится аутентификация отправителя с помощью технологии 3DSecure либо, в случае если карта отправителя не участвует в 3DSecure, то аутентификация выполняется с помощью AV-транзакции на нулевую сумму. В ответе на проведение платежа направляется:

- для карт, которые участвуют в 3DSecure
  код html страницы с автоматическим перенаправлением на ACS банка эмитента. В странице расположено поле ввода одноразового идентификатора.
  для карт, которые не участвуют в 3DSecure
  в поле otpJson возвращается структура для формирования страницы с запросом LOCATE кода. Описание структуры: в структуре передаются два параметра типа строка

```
{
        “url”:””,
        “otp”:””
}
```

На основании данных JSON структуры, необходимо создать форму реализующую передачу методом POST значения OTP, введенного пользователем, на указанный URL.

Пример формы:

```
<form method="POST" action="{{ $url }}">
        <input required name="otp" value="">
        <input type="submit" value="Continue">
</form>
```

Полученные страницы необходимо вывести пользователю на сайте партнера. После ввода идентификатора производится перенаправление на CallbackURL сервиса PayRun. После подтверждения проведения операции в системе PayRun производится перенаправление (redirect 302) пользователя на CallbackURL партнера который передан в параметре CallbackURL запроса.

Ссылка формируется формируется следующим образом:
CallbackURL?status=КОД
Где КОД - соответствует статусу обработки операции (см. раздел “Статус обработки операции”).

Тестовое списание/зачисление. Тестовые карты

Для тестирования операций списания и зачисления не среде разработки используется заглушка.

Для операций списания необходимо указывать любые валидные номера карт. Вместо реальной страницы 3ds, будет отдана тестовая страница с двумя кнопками: для успешного завершения операции и неуспешного, при этом будут вызваны необходимые колбеки и осуществлены перенаправления на необходимые страницы.

Для операций зачисления, для всех валидных номеров карт будет возвращен успех. Для получения неуспешного ответа на запрос зачисления, необходимо указать карту 4000000000000010.

Описание запросов

Карта-карта

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":10101,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции “счет-карта” помещаются в JSON структуру типа InitTransferC2C которая содержит следующие поля с именами:

```
{
   "TransferC2C":{
      "Sum":,
      "SenderCard":{
         "PAN":"",
         "ExpMon":"",
         "ExpYear":"",
         "CVV":""
      },
      "RecipientCard":{
         "PAN":""
      },
      "CallbackURL":""
   },
   "Transaction":{
      "TransactionID":"",
      "TerminalID":"",
      "DateTime":""
   }
}
```

TransferC2C - структура содержит данные об операции

Поле	Тип	Описание

Sum	Целое	Сумма операции в копейках (для платежа 1 грн передае 100).
SenderCard	Структура	Данные карты отправитель платежа
RecipientCard	Структура	Данные карты получателя платежа
CallbackURL	Строка	CallbackURL, вызывается после проведения операции н стороне PayRun

Transaction - структура содержит данные о транзакции

Поле	**Тип**	**Описание**
TransactionID	Строка	Уникальный идентификатор транзакции в системе партнера.
TerminalID	Строка	Уникальный идентификатор терминала в системе партнера.
DateTime	Строка	Дата и время проведения транзакции. Формат YYYYMMDD HH24:MM:SS

Составные структуры:

SenderCard - структура содержит данные банковской карты отправителя

Поле	**Тип**	**Описание**
PAN	Строка(14)	PAN карты
ExpMon	Строка(2)	Месяц срока окончания карты. Подлежит выравниванию до 2-х знаков символом ‘0’
ExpYear	Строка(2)	Последние 2 цифры года срока окончания карты. Подлежит выравниванию до 2-х знаков символом ‘0’
CVV	Строка(3)	CVV карты

RecipientCard - структура содержит данные кошелька.

Поле	**Тип**	**Описание**
PAN	Строка	PAN карты получателя платежа

Ответ

Операция успешно проведена:
{"Code":200,"Message":"operation ok","Data":{"OperationID": 11,"OperationStatus”: 10},"KeyAES":"","Sign":""}

Карта-кошелек

При проведении операции необходимо придерживаться использование типа операции согласно указанным в таблице правилам:

OperationType	Использование
10102	Операции до 14 000 грн для карты банков Украины за исключением карт Приватбанка
101021	Операции: ● свыше 14 000 грн для карты банков Украины, ● для любой суммы по картам Приватбанка
101022	Операции по картам зарубежных банков

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":10102 / 101021 / 101022,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции “карта-кошелек” помещаются в JSON структуру типа InitTransferC2W которая содержит следующие поля с именами:

```
{
   "Wallet":{
      "ID":"",
      "UserName":""
   },
   "TransferC2W":{
      "Sum":,
      "SenderCard":{
         "PAN":"",
         "ExpMon":"",
         "ExpYear":"",
         "CVV":""
         “GooglePay”:{
                  "data": <G-pay token в формате ecv2>,
                  "pub_key":
                  "MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEnHt5JtnLFm34dE8DRprNF
                  xOOmV8
                  nWUTwIRRU7KGk9gZcSDbMWeRL8kv2dJ0zgxX2n8shTvD60xGN3Z1wXGt
                  Hhw==",
                  },
      },
```

```
    "RecipientWallet":{
       "ID":"",
       "UserName":""
    },
    "CallbackURL": "",
    "FailedPage": "",
    "SuccessPage": ""
  },
  "Transaction":{
    "TransactionID":"",
    "TerminalID":"",
    "DateTime":""
  }
}
```

Wallet - структура содержит данные кошелька.

Поле	**Тип**	**Описание**
ID	Строка	Идентификатор кошелька итогового получателя средств системе PayRun (мерчанта) Идентификатором может выступать: 1. номер телефона 2. email 3. уникальный идентификатор клиента в системе партнера 4. номер карты клиента
UserName	Строка	Наименование клиента

TransferC2W - структура содержит данные об операции

Поле	**Тип**	**Описание**
Sum	Целое	Сумма операции в копейках (для платежа 1 грн передае 100).
SenderCard	Структура	Отправитель платежа
RecipientWallet	Структура	Кошелек получателя платежа при выполнении списания карты
CallbackURL	Строка	CallbackURL, вызывается после проведения операции на стороне PayRun

Transaction - структура содержит данные о транзакции

Поле	**Тип**	**Описание**
TransactionID	Строка	Уникальный идентификатор транзакции в системе партнера.
TerminalID	Строка	Уникальный идентификатор терминала в системе партнера.
DateTime	Строка	Дата и время проведения транзакции. Формат YYYYMMDD HH24:MM:SS

Составные структуры:

SenderCard - структура содержит данные банковской карты отправителя

Поле	Тип	Описание
PAN	Строка(14)	PAN карты
ExpMon	Строка(2)	Месяц срока окончания карты. Подлежит выравниванию до 2-х знаков символом ‘0’
ExpYear	Строка(2)	Последние 2 цифры года срока окончания карты. Подлежит выравниванию до 2-х знаков символом ‘0’
CVV	Строка(3)	CVV карты
GooglePay	Структура	Авторизационные данные платежа Google Pay(токен)

RecipientWallet - структура содержит данные кошелька.

Поле	Тип	Описание
ID	Строка	Идентификатор кошелька в системе PayRun Идентификатором может выступать: - номер телефона - email - уникальный идентификатор клиента в системе партнера - номер карты клиента
UserName	Строка	Наименование пользователя

Ответ

- запрос успешно принят и требует подтверждения 3ds:

{"Code":102,"Message":"need3ds","Data":{"OperationID":111, "3dsHtml": "<html страница в кодировке base64>","OperationStatus": 2},"KeyAES":"","Sign":""}

- запрос успешно принят и требует подтверждения OTP:

{"Code":102,"Message":"needOTP","Data":{"OperationID":111, "otpJson": "<JSON структура в виде строки>","OperationStatus": 3},"KeyAES":"","Sign":""}
{,"code": 102,"message": ""}

- операция успешно проведена:

{"Code":200,"Message":"done","Data":{"OperationID:555","OperationStatus”:10},"KeyAES":"","Sign":""}

- ошибка выполнения операции:

{"Code":200,"Message":"done","Data":{"OperationID":111, "OperationStatus":21, “Reason”:3},"KeyAES":"","Sign":""}

При наличии детализации причины отклонения операции, код причины указан в поле “Reason” (см. таблицу “Коды причин отклонения операций”)

Дальнейшая обработка OTP и 3ds проходят согласно разделу “Обработка запроса (CallbackURL)”

Кошелек-кошелек

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
"PartnerToken":"", "OperationType":10202, "Locale":""
}
```

“**Data**" - Все необходимые данные операции “кошелек-кошелек” помещаются в JSON структуру типа InitTransferW2W которая содержит следующие поля с именами:

```
{
 "TransferW2W":{
     "Sum":,
     "SenderWallet":{
        "ID":"",
        "UserName":""
     },
     "RecipientWallet":{
        "ID":"",
        "UserName":""
     },
  },
  "Transaction":{
     "TransactionID":"",
     "TerminalID":"",
     "DateTime":""
  }
}
```

TransferC2W - структура содержит данные об операции

Поле	**Тип**	**Описание**
Sum	Целое	Сумма операции в копейках (для платежа 1 грн передае 100).
SenderWallet	Структура	Данные кошелька отправителя платежа
RecipientWallet	Структура	Данные кошелька получателя платежа

Transaction - структура содержит данные о транзакции

Поле	**Тип**	**Описание**
TransactionID	Строка	Уникальный идентификатор транзакции в системе партнера.
TerminalID	Строка	Уникальный идентификатор терминала в системе партнера.
DateTime	Строка	Дата и время проведения транзакции. Формат YYYYMMDD HH24:MM:SS

SenderWallet и RecipientWallet - структура содержит данные кошелька.

Поле	Тип	Описание
ID	Строка	Идентификатор кошелька в системе PayRun Идентификатором может выступать: - номер телефона (в формате “380123456789”, либ “0123456789”) - email
UserName	Строка	Наименование пользователя Допускается пустое значение

Ответ

Операция успешно проведена:
{"Code":200,"Message":"operation ok","Data":{"OperationID": 11,"OperationStatus”: 10},"KeyAES":"","Sign":""}

Счет-карта

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":10301/10311,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции “счет-карта” помещаются в JSON структуру типа InitTransferA2C которая содержит следующие поля с именами:

```
{
   "TransferA2C":{
      "Sum":,
      "Phone":"",
      "RecipientCard":{
         "PAN":""
      },
      "FirstName":"",
      "LastName":"",
      "MiddleName":"",
   },
   "Transaction":{
      "TransactionID":"",
```

```
        "TerminalID":"",
        "DateTime":""
    }
}
```

Счет отправителя в запросе не участвует. Он настраивается в системе PayRun для операции согласно договору с партнером.

TransferA2C - структура содержит данные об операции

Поле	**Тип**	**Описание**
Sum	Целое	Сумма операции в копейках (для платежа 1 грн передае 100).
Phone	*Строка*	Телефон клиента (в формате “380xxxxxxxxx”). *Не обязательное поле.*
RecipientCard	Структура	Данные карты получателя платежа
FirstName	Строка	Имя клиента
MiddleName	Строка	Отчество клиента
LastName	Строка	Фамилия клиента

Transaction - структура содержит данные о транзакции

Поле	**Тип**	**Описание**
TransactionID	Строка	Уникальный идентификатор транзакции в системе партнера.
TerminalID	Строка	Уникальный идентификатор терминала в системе партнера.
DateTime	Строка	Дата и время проведения транзакции. Формат YYYYMMDD HH24:MM:SS

Ответ

Операция успешно проведена:

{"Code":200,"Message":"operation ok","Data":{"OperationID": 11,"OperationStatus”: 10},"KeyAES":"","Sign":""}

Кошелек-UUID

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
"PartnerToken":"", "OperationType":10304, "Locale":""
}
```

“**Data**" - Все необходимые данные операции “кошелек-UUID” помещаются в JSON структуру типа InitTransferW2U которая содержит следующие поля с именами:

```
{
   "Sum":,
   "Recipient":"",
   "Transaction":{
      "TransactionID":"",
      "TerminalID":"",
      "DateTime":""
   }
}
```

Кошелек отправителя в запросе не участвует. Он настраивается в системе PayRun для операции согласно договору с партнером.

InitTransferW2U - структура содержит данные об операции

Поле	**Тип**	**Описание**
Sum	Целое	Сумма операции в копейках (для платежа 1 грн передае 100).
Recipient	Строка	UUID карты получателя платежа
Transaction	Структура	Структура данных о транзакции

Transaction - структура содержит данные о транзакции

Поле	**Тип**	**Описание**
TransactionID	Строка	Уникальный идентификатор транзакции в системе партнера.
TerminalID	Строка	Уникальный идентификатор терминала в системе партнера.
DateTime	Строка	Дата и время проведения транзакции. Формат YYYYMMDD HH24:MM:SS

Ответ

Операция успешно проведена:
{"Code":200,"Message":"operation ok","Data":{"OperationID": 11,"OperationStatus”: 10},"KeyAES":"","Sign":""}

Пакетное перечисление

Запрос

"Partner" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":10400,
   "Locale":""
}
```

“Data" - Все необходимые данные операции пакетного перечисления помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "InitTransferW2T":{
      "TransferW2T":{
         "Sum":,
         "SenderWallet":{
            "ID":"",
            "UserName":""
         },
         "RecipientToken":""
      },
      "Transaction":{
         "TransactionID":"",
         "TerminalID":"",
         "DateTime":""
      }
   },
   "Bunch":[
   {
      "TransferW2W":{
         "Sum":,
         "RecipientWallet":{
            "ID":"",
            "UserName":""
         }
      },
      "Transaction":{
         "TransactionID":"",
         "TerminalID":"",
         "DateTime":""}
   },
   …]
}
```

TransferW2T - структура содержит данные об операции

Поле	**Тип**	**Описание**
Sum	Целое	Сумма операции в копейках (для платежа 1 грн передае 100).
SenderWallet	Структура	Данные кошелька отправителя платежа
RecipientToken	Строка	Токен карты получателя платежа. В системе PayRun получатель платежа определяется по токенизированной и привязанной к кошельку карте

TransferW2W - структура содержит данные об операции

Поле	**Тип**	**Описание**
Sum	Целое	Сумма операции в копейках (для платежа 1 грн передае 100).
RecipientWallet	Структура	Данные кошелька получателя платежа

Transaction - структура содержит данные о транзакции

Поле	Тип	Описание
TransactionID	Строка	Уникальный идентификатор транзакции в системе партнера.
TerminalID	Строка	Уникальный идентификатор терминала в системе партнера.
DateTime	Строка	Дата и время проведения транзакции. Формат YYYYMMDD HH24:MM:SS

Пример. Рассмотрим формирование операций в системе на основании приведенного ниже запроса.

```
{
   "InitTransferW2T":{
      "TransferW2T":{
         "Sum":1000,
         "SenderWallet":{
            "ID":"",
            "UserName":"Партнер 1"
            },
         "RecipientToken":"токен карты клиента"
      },
      "Transaction":{
         "TransactionID":"",
```

```
        "TerminalID":"",
        "DateTime":""
      }
  },
  "Bunch":[
  {
      "TransferW2W":{
        "Sum":100,
        "RecipientWallet":{
          "ID":"",
          "UserName":"Партнер 2"
        }
      },
      "Transaction":{
        "TransactionID":"",
        "TerminalID":"",
        "DateTime":""
      }
  },
  {
      "TransferW2W":{
        "Sum":300,
        "RecipientWallet":{
          "ID":"",
          "UserName":"Партнер 3"
        }
      },
       "Transaction":{
        "TransactionID":"",
       "TerminalID":"",
        "DateTime":""
      }
  }
]
```

}

Схема проведения платежей:

№	Отправитель	Получатель	Сумма	Описание
1	Кошелек: Партнер 1	Кошелек пользователя привязанного к токен карты клиента	Сумма всех операций в запросе 1400	Списание средс с кошелька партнера на кошелек клиента
2	Кошелек пользовате привязанного к токен карты клиента	Карта пользователя	1000	Перечисление средств на карту пользователя
3		Кошелек: Партнер 2	100	Перечисление средств партнер 2
4		Кошелек: Партнер 3	300	Перечисление средств партнер 3

Ответ

Операция успешно проведена:
{"Code":200,"Message":"operation ok","Data":{"OperationID": 11,"OperationStatus”: 10},"KeyAES":"","Sign":""}

Операция успешно проведена:
{"Code":200,"Message":"","Data":{"OperationID": 11,"OperationStatus”: 21, “Reason”: 3},"KeyAES":"","Sign":""}

Пакетное перечисление на карту

Запрос

"Partner" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":10401,
   "Locale":""
}
```

“Data" - Все необходимые данные операции пакетного перечисления помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "WalletID":{
      "TransferW2T":{
         "Sum":,
         "SenderWallet":{
            "ID":"",
            "UserName":""
         },
         "Pan":"",
         "WalletID":""
      },
      "Transaction":{
         "TransactionID":"",
         "TerminalID":"",
         "DateTime":""
      }
   },
   "Bunch":[
   {
      "TransferW2W":{
         "Sum":,
         "RecipientWallet":{
            "ID":"",
            "UserName":""
         }
      },
      "Transaction":{
         "TransactionID":"",
         "TerminalID":"",
         "DateTime":""
      }
   },
   …]
```

}

TransferW2T - структура содержит данные об операции

Поле	Тип	Описание
Sum	Целое	Сумма операции в копейках (для платежа 1 грн передае 100).
SenderWallet	Структура	Данные кошелька отправителя платежа
Pan	Строка	PAN карты клиента для зачисление средств
WalletID		Идентификатор кошелька итогового получателя средств системе PayRun (клиента) Идентификатором может выступать: 1. номер телефона 2. email

TransferW2W - структура содержит данные об операции

Поле	Тип	Описание
Sum	Целое	Сумма операции в копейках (для платежа 1 грн передае 100).
RecipientWallet	Структура	Данные кошелька получателя платежа

Transaction - структура содержит данные о транзакции

Поле	Тип	Описание
TransactionID	Строка	Уникальный идентификатор транзакции в системе партнера.
TerminalID	Строка	Уникальный идентификатор терминала в системе партнера.
DateTime	Строка	Дата и время проведения транзакции. Формат YYYYMMDD HH24:MM:SS

Пример. Рассмотрим формирование операций в системе на основании приведенного ниже запроса.


```
{
   "InitTransferW2T":{
      "TransferW2T":{
         "Sum":1000,
         "SenderWallet":{
            "ID":"",
            "UserName":"Партнер 1"
         },
```

```
        "Pan":"1234567890123456",
        "WalletID":"0993216547"
    },
    "Transaction":{ "TransactionID":"",
        "TerminalID":"",
        "DateTime":""
    }
},
"Bunch":[
{
    "TransferW2W":{
        "Sum":100,
        "RecipientWallet":{
            "ID":"",
            "UserName":"Партнер 2"
        }
    },
    "Transaction":{
        "TransactionID":"",
        "TerminalID":"",
        "DateTime":""
    }
},
{
    "TransferW2W":{
        "Sum":300,
        "RecipientWallet":{
            "ID":"",
            "UserName":"Партнер 3"
        }
    },
    "Transaction":{
        "TransactionID":"",
        "TerminalID":"",
        "DateTime":""
        }
    }
```

]

}

Схема проведения платежей:

№	Отправитель	Получатель	Сумма	Описание
1	Кошелек: Партнер 1	Кошелек пользователя привязанного к токен карты клиента	Сумма всех операций в запросе 1400	Списание средс с кошелька партнера на кошелек клиента
2	Кошелек пользовате привязанного к токен карты клиента	Карта пользователя	1000	Перечисление средств на карту пользователя
3		Кошелек: Партнер 2	100	Перечисление средств партнер 2
4		Кошелек: Партнер 3	300	Перечисление средств партнер 3

Ответ

Операция успешно проведена:
{"Code":200,"Message":"done","Data":{"OperationID": 11,"OperationStatus”: 10},"KeyAES":"","Sign":""}

Операция завершилась неудачей:
{"Code":200,"Message":"fail","Data":{"OperationID": 11,"OperationStatus”: 21, “Reason”: 3},"KeyAES":"","Sign":""}

Перечисление с карты по токену

Запрос

При проведении операции необходимо придерживаться использование типа операции согласно указанным в таблице правилам:

OperationType	Использование
104021	Операции: ● до 14 000 грн для карты банков Украины
104022	Операции: ● свыше 14 000 грн для карты банков Украины
10402	Операции с автоматическим выбором шлюза

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":104021 / 104022 / 10402,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "Phone":"",
   "Sum":,
   "ClientToken":"",
   "Action":"Prepare” / “Pay",
   "SuccessCallback":"",
   "FailedCallback":"",
   "Transaction":{
        "TransactionID":"",
        "TerminalID":"",
        "DateTime":""
   }
}
```

Данные операции запроса баланса

Поле	Тип	Описание
Phone	Строка	Телефон клиента (в формате “380xxxxxxxxx”)
Sum	Целое	Сумма в копейках

ClientToken	Строка	Токен карты клиента
Action	Строка	Тип запроса (не обязательное поле: по умолчанию значение “Prepare”). Возможные значения: Prepare - расчет комиссии Pay - проведение оплаты

SuccessCallback	Строка	URL который вызывается в случае успешного проведени операции на стороне PayRun
FailedCallback	Строка	URL который вызывается в случае ошибки проведения операции на стороне PayRun

Ответ

Пример ответа при "Action":"Prepare":

{"Code":102,"Message":"done","Data":{"Sum":500,“Fee”:50,”TotalSum”:550,"OperationStatus”:7},"KeyAES":"","Sign":""}

В поле “Data” присылаются данные значений сумм платежа и комиссии

Поле	**Тип**	**Описание**
Sum	Целое	Запрошенная сумма перевода в копейках
Fee	Целое	Сумма комиссии в копейках
TotalSum	Целое	Общая сумма перевода в копейках

После этого, для проведения платежа необходимо повторить запрос с теми же данными, но поле "Action" должно иметь значение "Pay".

Пример ответа при "Action":"Pay":

- запрос успешно принят и требует подтверждения 3ds:

{"Code":102,"Message":"need3ds","Data":{"OperationID":111, "3dsHtml": "<html страница в кодировке base64>","OperationStatus": 2},"KeyAES":"","Sign":""}

- запрос успешно принят и требует подтверждения OTP:

{"Code":102,"Message":"needOTP","Data":{"OperationID":111, "otpJson": "<JSON структура в виде строки>","OperationStatus": 3},"KeyAES":"","Sign":""}
{,"code": 102,"message": ""}

- операция успешно проведена:

{"Code":200,"Message":"done","Data":{"OperationID:555","OperationStatus”:10},"KeyAES":"","Sign":""}

- ошибка выполнения операции:

{"Code":200,"Message":"done","Data":{"OperationID":111, "OperationStatus":21, “Reason”:3},"KeyAES":"","Sign":""}

При наличии детализации причины отклонения операции, код причины указан в поле “Reason” (см. таблицу “Коды причин отклонения операций”)
Дальнейшая обработка OTP и 3ds проходят согласно разделу “Обработка запроса (CallbackURL)”

Получение ссылки платежного виджета

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":10005,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
  "PayType": "2",
  "Phone": "",
  "Email": "",
  "ClientIP": "",
  "PanSuffix": "",
  "Signature": "",
  "AccountID": "",
  "Sum": "",
  "FirstName": "",
  "LastName": "",
  "MiddleName": "",
  "Goods": [
    {"Amount": 1230,
     "Count": 1,
     "Name": "Утюг",
     "Description": "Утюг электрический"
    },
    {"Amount": 7340,
     "Count": 1,
     "Name": "Доска",
     "Description": "Доска для глажки"
    }
  ],
  "Transaction": {
    "TransactionID": "",
    "TerminalID": "",
    "DateTime": ""
  }
}
```

Данные операции запроса платежного виджета

Поле	**Тип**	**Описание**
PayType	Строка	Указание типа платежного метода. Возможные варианты:
		“0/пустое значения/отсутсвие параметра” - оплата через c2a (списание с карты через p2p перевод); “1” - оплата через шлюз интернет эквайринга; “2” - отложенный платеж.* “3” - оплата через сторонний виджет “4” - оплата по реквизитам*** “5” - отложенный платеж по реквизитам**** “10” - оплата в кредит (оформление кредита)
*Phone***	*Строка*	*Телефон клиента (в формате “380xxxxxxxxx”)*
*Email***	*Строка*	*Email клиента*
ClientIP	Строка	IP адрес с которого клиент инициирует операцию
PanSuffix	Строка	последние 4 цифры номера карты в случае если была предварительная верификация карты
Signature	Строка	Буквенный код валюты (“XPYUAH”, “LTC”, “NMC”, “XRP”...) согласно https://jsons.info/signatures/currencies
AccountID	Строка	Номер криптовалютного кошелька пользователя / айди счета в платежной системе / айди перевода / номер счета онлайн банкинга или биржи / айди денежного перевода / айди чека или ваучера
Sum	Строка	Сумма валюты операции обмена (количество покупаемой валюты, например: “0.0025”)
FirstName	Строка	Имя клиента
MiddleName	Строка	Отчество клиента
LastName	Строка	Фамилия клиента

BrowserFingerprint	Строка	Идентификатор браузера https://m.habr.com/ru/company/oleg-bunin/blog/3212 94/ https://github.com/valve
Goods	Массив	Массив с описанием товара. Обязателен для PayType = 10 {"Amount": 1230, -- стоимость товара в коп "Count": 1, -- количество единиц товара "Name": "Утюг", -- наименование товара "Description": "Утюг электрический" -- описание товара }

*при совершении отложенного платежа, необходим последующий запрос подтверждения списания (операция 102070221), либо отмена (102070222). **В случае отсутствия подтверждения списания, спустя время определенное в договоре с партнером, списание будет автоматически отменено.**
**при указании данных полей, будет неявно выполнен запрос check двухфакторного взаимодействия, и отдана ссылка на виджет для ввода карточных данных
***при запросе виджета для оплаты по реквизитам (PayType = 4 или 5), на запрос check необходимо отдать данные реквизитов для оплаты (см. “Поля дополнительных данных структуры info”)
****при запросе виджета для отложенной оплаты по реквизитам (PayType = 5), необходим последующий запрос подтверждения списания (операция 102070251), либо отмена (102070252). **В случае отсутствия подтверждения списания, спустя время, определенное в договоре с партнером, списание будет автоматически отменено.**

Взаимодействие с родительской страницей:
В браузере пользователя платежный виджет может взаимодействовать с родительским документом путем создания события.
Событие "close" инициируется нажатием соответствующей кнопки пользователем или по таймеру после успешного завершения платежа.

В родительском документе это событие можно перехватить.
Напр. при помощи JS так:

```
window.addEventListener("message", function(e) {

    if(event.data.event_id === 'close'){
        // do something ..
        }
    }, false);
```

Пример:

Запрос

{"Partner":{"PartnerToken":"f3a5fa5a-6e28-4113-99f6-72b32eada0cc","OperationType":10005},"Data":"{\"Transaction\":{\"TransactionID\":\"45913135c6e2be0cbf58\",\"TerminalID\":\"0\",\"DateTime\":\"20190221 06:41:04\"}}","KeyAES":"","Sign":""}

Ответ

Успешный ответ содержит ссылку в виде строки с именем "URI" в поле “Data”

Поле	**Тип**	**Описание**
URI	Строка	Ссылка на платежный виджет
uuid	Строка	Уникальный идентификатор связывающий запросы виджета/check/pay

Операция успешно проведена:
{"Code":200,"Message":"done","Data":{"OperationID":11,"OperationStatus”:10,"URI":"https://stage-mapi.payrun.online/ru/frame/widget/f3cd72b6-e1ea-406f-9b44-a9b93b401b7f","uuid":"f3cd72 b6-e1ea-406f-9b44-a9b93b401b7f"},"KeyAES":"","Sign":""}

Протокол двухфакторного взаимодействия

Используется для быстрого доступа к сервисам партнера через платежный виджет. Для этого, партнер должен реализовать поддержку запросов и ответов согласно данному протоколу.
Операция состоит из двух типов запросов:
check -- получение данных от партнера для операции по клиенту
pay -- непосредственно информирование партнера об успешной операции

Партнер предоставляет URL-ы для осуществления запросов check и pay.
Оба запроса осуществляются методом GET и содержат строку параметров.
В ответ партнер отправляет данные, тип значения которых -- строки.

Параметры запроса *check*

Параметр	**Описание**	**Обязательность**	**Пример**
command	Тип запроса	да	check
txn_id	Идентификатор транзакц	да	321455
uuid	Уникальный идентификатор связывающий запросы виджета/check/pay	да	f3cd72b6-e1ea-406f-944-a9b93b401b7f
account	Идентификатор клиента в системе партнера (телефон або email)	да	380638754213
sum	Сумма операции в копейках	нет	100
pay_type	Идентификатор услуги	да	1
locale	Локаль виджета (соответствует переданно локали при запросе виджета 10005)	да	uk

Ответ партнера должен быть в формате JSON, кодировке UTF-8 и иметь следующие данные

Данные ответа на запрос *check*

Параметр	Описание	Обязательность	Пример
txn_id	Идентификатор транзакц	да	321455
result	Кодовое значение результата запроса	да	10*
message	Текстовое сообщение с описанием результата	Нет	Ок
info	Структура содержащая дополнительные данные	Нет	

*значение кода ответа, должно соответствовать таблице “Статус обработки операции”

В случае вызова виджета для оплаты по реквизитам, содержит структуру данных реквизитов в виде JSON строки

Поля дополнительных данных структуры info

Параметр	Тип	Описание	Обязательность	Пример
client_name	Строка	ФИО клиента	нет	Кіндратенко Генадій Йосипович
contract_number	Строка	Номер заказа в системе партнера	нет	98-00ФАВ
max_sum	Строка	Максимально доступная сумма платежа в копейках. Может быть изменена клиентом.	да/нет*	3189
fixed_sum	Строка	Фиксированная сумма платежа в копейках. Не может быть изменен клиентом.	нет/да*	3189
account	Структура	Данные реквизитов**	Да при PayType=4/5	

* один из параметров обязателен
**структура данных реквизитов

```
{
        "EDRPOU":"",
        "Account":"",
        "MFO":"",
        "Purpose":"",
        "SenderName":"",
        "RecipientName":""
}
```

Поле	Тип	Описание
EDRPOU	Строка	ЕДРПОУ получателя
Account	Строка	Расчетный счет получателя или IBAN
MFO	Строка	МФО
Purpose	Строка	Описание назначения платежа
SenderName	Строка	ФИО отправителя
RecipientName	Строка	ФИО/наименование получателя

Пример запроса:
https://partner.host/protocol?command=check&txn_id=321455&uuid=f3cd72b6-e1ea-406f-9b44-a9b93b401b7f&account=380638754213&sum=100&pay_type=1&locale=uk

Пример ответа:

```
{
        "txn_id":"321455",
        "result":"10",
        "message":"Ok",
        "info":{
                "client_name":"Кіндратенко Генадій Йосипович",
                "contract_number":"98-00ФАВ",
                "fixed_sum":"3189"
        }
}
```

Параметры запроса pay

Параметр	Описание	Обязательность**	Пример
command	Тип запроса	да	pay
txn_id*	Уникальный идентификатор транзакции	да	321455
uuid	Уникальный идентификатор связывающий запросы виджета/check/pay	да	f3cd72b6-e1ea-406f-944-a9b93b401b7f
account	Идентификатор клиента в системе партнера	да	380638754213
sum	Сумма операции в копейках	да	100
pay_type	Идентификатор услуги	да	1
txn_date	Время проведения операции	да	20190301_180233
card_token	токен карты использованной для оплаты в данном виджете	нет	c0bbb6318f7c4422b987bc97241452c0d925

			cdd2a1d
locale	Локаль виджета (соответствует переданной локали при запросе виджета 10005)	да	uk

*при получении запроса pay с txn_id который ранее уже передавался, запрос следует считать повторным, при этом не должна генерироваться новая транзакция.
**отправка необязательных параметров регулируется договорными отношениями

Параметры ответа на запрос pay

Параметр	**Тип**	**Описание**	**Обязательность**	**Пример**
result	Строка	Кодовое значение результата запроса	да	10
txn_id	Строка	Идентификатор транзакции	да	321456
message	Строка	Текстовое сообщение с описанием результата	Нет	Ок
date_time	Строка	Время проведения операции в системе партнера	да	20190301180 33

Пример запроса:
https://partner.host/protocol?command=pay&txn_id=321456&uuid=f3cd72b6-e1ea-406f-9b44-a9b93b401b7f&account=380638754213&sum=100&pay_type=1&date_time=20190301_180233&locale=uk

Пример ответа:

```
{
        "txn_id":"321456",
        "result":"10",
        "message":"Done",
        "date_time":"20190301_180833"
}
```

Запрос предавторизации при отложенной оплате

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":10207022,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "PayType":"",
   "ClientIP":"",
   “Recipient”:””,
   "Transaction":{
        "TransactionID":"",
        "TerminalID":"",
        "DateTime":""
   },
   “TransferC2W”:{
     “Sum”:,
     “SenderCard” : {
     “CVV”: “”,
     “PAN”:””,
     “ExpMon”: “”,
     “ExpYear”:””,
     },
     “RecipientWallet”:{
     “ID”:””,
     “UserName”:””
     }
   }
}
```

Данные операции запроса баланса

Поле	**Тип**	**Описание**
PayType	Строка	Указание типа платежного метода. Возможные варианты: “2” - отложенный платеж.*
ClientIP	Строка	IP адрес с которого клиент инициирует операцию
Recipient	Строка	UUID карты получателя платежа

TransferC2W - структура содержит данные об операции

Поле	**Тип**	**Описание**
Sum	Целое	Сумма операции в копейках (для платежа 1 грн передае 100).
SenderCard	Структура	Отправитель платежа
RecipientWallet	Структура	Кошелек получателя платежа при выполнении списания карты

SenderCard - структура содержит данные банковской карты отправителя

Поле	Тип	Описание
PAN	Строка(14)	PAN карты
ExpMon	Строка(2)	Месяц срока окончания карты. Подлежит выравниванию до 2-х знаков символом ‘0’
ExpYear	Строка(2)	Последние 2 цифры года срока окончания карты. Подлежит выравниванию до 2-х знаков символом ‘0’
CVV	Строка(3)	CVV карты

RecipientWallet - структура содержит данные кошелька.

Поле	Тип	Описание
ID	Строка	Идентификатор кошелька в системе PayRun Идентификатором может выступать: - номер телефона - email - уникальный идентификатор клиента в системе партнера - номер карты клиента
UserName	Строка	Наименование пользователя

Ответ

{"Code":102,"Message":"needACS","Data":{"3dsHtml":"<html страница в кодировке base64","OperationID":111,"OperationStatus":2},"KeyAES":"","Sign":""}

Запрос подтверждения/отмены отложенной оплаты

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":102070221 / 102070222 / 102070251 / 102070252,
"Locale":""
   }
```

OperationType	Использование
102070221	Подтверждение отложенной оплаты

102070222	Отмена отложенной оплаты
102070251	Подтверждение отложенной оплаты по реквизитам
102070252	Отмена отложенной оплаты по реквизитам

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "PreauthorizationID":"",
   "Sum":,
   "Transaction":{
        "TransactionID":"",
        "TerminalID":"",
        "DateTime":""
   }
}
```

Данные операции запроса баланса

Поле	**Тип**	**Описание**
PreauthorizationID	Строка	Значение параметра *txn_id* полученное в запросе PAY
Sum*	Целое	Сумма поставторизации в копейках. Может отличаться как в большую, так и в меньшую сторону от суммы предавторизации не более чем на 15% (процент оговаривается индивидуально)

* передается только в запросе подтверждения оплаты

Ответ

{"Code":200,"Message":"Done","Data":{"OperationID": 11,"OperationStatus”: 10},"KeyAES":"","Sign":""}

Запрос погашения платежа

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":10100,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "Phone":"",
   "Sum":,
   "ClientToken":"",
   "SuccessCallback":"",
   "FailedCallback":""
}
```

Данные операции запроса баланса

Поле	**Тип**	**Описание**
Phone	Строка	Телефон клиента (в формате “380xxxxxxxxx”)
Sum	Целое	Сумма в копейках
ClientToken	Строка	Токен карты клиента
SuccessCallback	Строка	URL который вызывается в случае успешного проведени операции на стороне PayRun
FailedCallback	Строка	URL который вызывается в случае ошибки проведения операции на стороне PayRun

Ответ

{"Code":200,"Message":"done","Data":{"PayHash":”7c09daee98974402d64ab68c6d74c075c5aa588d0814801a266f909020aea12”},"KeyAES":"","Sign":""}

Отмена платежа (возврат)

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":30201,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру типа “InitRefund” которая содержит следующие поля с именами:

```
{
   "OperationID": ,
   "Transaction":{
      "TransactionID":"",
      "TerminalID":"",
      "DateTime":""
   }
}
```

InitRefund- структура содержит данные об операции

Поле	**Тип**	**Описание**
OperationID	Целое	Идентификатор операции в системе PayRun, которая требует отмены

Ответ

Операция успешно проведена:
{"Code":200,"Message":"operation ok","Data":{"OperationID": 11,"OperationStatus”: 10},"KeyAES":"","Sign":""}

Отмена поставторизации (возврат)

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":30202,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру типа “InitRefund” которая содержит следующие поля с именами:

```
{
   "OperationID": ,
   "Transaction":{
      "TransactionID":"",
      "TerminalID":"",
      "DateTime":""
   }
}
```

InitRefund- структура содержит данные об операции

Поле	**Тип**	**Описание**
OperationID	Целое	Идентификатор операции в системе PayRun, полученный в ответе на запрос поставторизации

Ответ

Операция успешно проведена:
{"Code":200,"Message":"operation ok","Data":{"OperationID": 11,"OperationStatus”: 10},"KeyAES":"","Sign":""}

Отмена (возврат) платежа совершенного через виджет

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":30203,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "TxnID": ,
   "Transaction":{
      "TransactionID":"",
      "TerminalID":"",
      "DateTime":""
   }
```

```
}
```

Поле	Тип	Описание
TxnID	Целое	Значение параметра *txn_id* полученное в запросе PAY

Ответ

Операция успешно проведена:
{"Code":200,"Message":"operation ok","Data":{"OperationID": 11,"OperationStatus”: 10},"KeyAES":"","Sign":""}

Получение данных карты по номеру телефона

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":20400 / 20410,
   "Locale":""
}
```

OperationType	Использование
20400	Получение данных последней привязанной карты
20410	Получение данных всех привязанных карт

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "Phone":""
}
```

Данные операции запроса баланса

Поле	Тип	Описание
Phone	Строка	Номер телефона пользователя (в формате “380xxxxxxxx”)

Ответ

Данные ответа приходят в виде:

- JSON структуры с именем "token" в поле “Data” на 20400
- массива JSON структур с именем "tokens" в поле “Data” на 20410

Поле	Тип	Описание
Token	Строка	Идентификатор карты в системе PayRun
MaskPan	Строка	Маскированный PAN карты
Bank	Строка	Имя банка эмитента
PaySys	Строка	Международная платежная система

Пример успешного ответа на запрос 20400:

{"Code":200,"Message":"done","Data":{"token":"{\"Token\":\"7a8250a2327737abe5f2febd3f645afcaf456742b296344ec29c70b2a3abe744\",\"MaskPan\":\"419811******2503\",\"Bank\":\"Bank\",\"PaySys\":\"VISA\"}"},"KeyAES":"","Sign":""}

Пример успешного ответа на запрос 20410:

{"Code":200,"Message":"done","Data":{"OperationStatus":10,"tokens":"[{\"Token\":\"20a9c01be1c24feb2a8f77385a2a469b90a9258bdd6b916cb6f03ea294257b31\",\"MaskPan\":\"516936******0708\",\"Bank\":\"PRIVATBANK\",\"PaySys\":\"MasterCard (IPM)\"},{\"Token\":\"e85785597108eb24814175906994df57157a351676bfd552dfc2f3b8efbf279f\",\"MaskPan\":\"516875******6358\",\"Bank\":\"PRIVATBANK\",\"PaySys\":\"MasterCard (IPM)\"},{\"Token\":\"73921e40843c4035296cfb9e810e9e353476bbe117d91591205e24ad639b1a2e\",\"MaskPan\":\"516936******0925\",\"Bank\":\"PRIVATBANK\",\"PaySys\":\"MasterCard (IPM)\"},{\"Token\":\"48c71533af0397ec583d7d4dd9a4865f559b15e007ccafe396eb67c5e2649d3e\",\"MaskPan\":\"444111******8660\",\"Bank\":\"Bank\",\"PaySys\":\"Card\"}]"},"KeyAES":"","Sign":""}

Получение ссылки виджета для выплаты

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
    "PartnerToken":"",
    "OperationType":10007,
    "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
  "Phone": "",
  "Email": "",
  "ClientIP": "",
  "Sum": "",
  "FirstName": "",
  "LastName": "",
  "MiddleName": "",
  "Purpose": "",
  "BrowserFingerprint": "",
  "Transaction": {
    "TransactionID": "",
    "TerminalID": "",
    "DateTime": ""
  }
}
```

Данные операции запроса платежного виджета

Поле	Тип	Описание
Phone**	Строка	Телефон клиента (в формате “380xxxxxxxxx”)
Email**	Строка	Email клиента
ClientIP	Строка	IP адрес с которого клиент инициирует операцию
Sum	Строка	Сумма валюты операции обмена (количество покупаемой валюты, например: “0.0025”)
FirstName	Строка	Имя клиента
MiddleName	Строка	Отчество клиента
LastName	Строка	Фамилия клиента

BrowserFingerprint	Строка	Идентификатор браузера https://m.habr.com/ru/company/oleg-bunin/blog/3212 94/ https://github.com/valve
Purpose	Строка	Назначение платежа

Взаимодействие с родительской страницей:
В браузере пользователя виджет может взаимодействовать с родительским документом путем создания события.
Событие "close" инициируется нажатием соответствующей кнопки пользователем или по таймеру после успешного завершения платежа.

В родительском документе это событие можно перехватить. Напр. при помощи JS так:
window.addEventListener("message", function(e) {

if(event.data.event_id === 'close'){
// do something ..
}
}, false);

Пример:

Ответ

Успешный ответ содержит ссылку в виде строки с именем "URI" в поле “Data”

Поле	Тип	Описание
URI	Строка	Ссылка на платежный виджет
uuid	Строка	Уникальный идентификатор

Операция успешно проведена:

{"Code":200,"Message":"done","Data":{"OperationID":11,"OperationStatus”:10,"URI":"https://stage-mapi.payrun.online/ru/frame/widget/f3cd72b6-e1ea-406f-9b44-a9b93b401b7f","uuid":"f3cd72 b6-e1ea-406f-9b44-a9b93b401b7f"},"KeyAES":"","Sign":""}

После завершения выплаты система формирует postback на указанный URI партнера.

Пример запроса:
https://partner.host/protocol?command=pay&txn_id=321456&uuid=f3cd72b6-e1ea-406f-9b44-a9b93b401b7f&account=380638754213&sum=100&date_time=20190301_180233

Пример ответа:

```
{
        "txn_id":"321456",
        "result":"10",
        "message":"Done",
        "date_time":"20190301_180833"
}
```

Баланс

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":20001,
   "Locale":""
```

```
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "ID":""
}
```

Данные операции запроса баланса

Поле	**Тип**	**Описание**
ID	Строка	Идентификатор кошелька в системе PayRun

Ответ

Баланс по счету (в нешифрованном виде)

{"Code":200,"Message":"done","Data":{"decodedData":"{\"ID\":\"UAH897589401\",\"Sum\":50000,\"DateTime\":\"20180510 13:23:19\"}"},"KeyAES":"","Sign":""}

Выписка по кошельку

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":20002,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "ID":""
   "DateStart":"",
   "DateFinish":""
}
```

Данные операции запроса выписки

Поле	**Тип**	**Описание**
ID	Строка	Идентификатор кошелька в системе PayRun (при пустом значении, выписка отображается по всем кошелькам партнера)

DateStart	Строка	Начальная дата периода выписки в формате YYYY-MM-DD 24HH:MM:SS
DateFinish	Строка	Конечная дата периода выписки в формате YYYY-MM-D 24HH:MM:SS

Ответ

Данные ответа приходят в виде JSON структуры с именем "decodedData" в поле “Data”

Поле	**Тип**	**Описание**
DateTime	Строка	Дата/время проведения операции
SenderID	Строка	Идентификатор отправителя
RecipientID	Строка	Идентификатор получателя
Sum	Строка	Сумма операции
State	Строка	Статус операции*
Extterminalid	Строка	Идентификатор терминала в системе партнера
Exttransactionid	Строка	Идентификатор транзакции в системе партнера

*Возможные статусы операции

Статус	**Финальность**	**Результат**
Created	нет	Создана
Processed	нет	В процессе
Done	да	Проведена
Declined	да	Не проведена
Canceled	да	Отменена

Пример выписки по кошельку (в нешифрованном виде):
{"Code":200,"Message":"done","Data":{"decodedData":"[{\"DateTime\":\"2018-08-30T11:59:56+03:00\",\"SenderID\":\"UAH703758543\",\"RecipientID\":\"UAH271057865\",\"Sum\":100,\"State\":\"Done\",\"Extterminalid\":\"1\",\"Exttransactionid\":\"14588\"},{\"DateTime\":\"2018-08-30T11:50:47+03:00\",\"SenderID\":\"UAH703758543\",\"RecipientID\":\"UAH271057865\",\"Sum\":100,\"State\":\"Done\",\"Extterminalid\":\"1\",\"Exttransactionid\":\"14584\"}]","partnerToken":"4348501a-b73a-4ea5-8e12-629cec3a97a2"},"KeyAES":"","Sign":""}

Эмиссия

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":30001,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "ID":"", // идентификатор кошелька
   "Sum": //сумма эмиссии в копейках
}
```

Данные операции запроса эмиссии/погашения

Поле	**Тип**	**Описание**
ID	Строка	Идентификатор кошелька в системе PayRun
Sum	Целое	Сумма эмиссии/погашения в копейках

Ответ

Операция успешно проведена:
{"Code":200,"Message":"operation ok","Data":{"OperationID: 11","OperationStatus: 10"},"KeyAES":"","Sign":""}

Погашение

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":30002,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "ID":"", // идентификатор кошелька
   "Sum": //сумма погашения в копейках
}
```

Данные операции запроса эмиссии/погашения

Поле	**Тип**	**Описание**

ID	Строка	Идентификатор кошелька в системе PayRun
Sum	Целое	Сумма эмиссии/погашения в копейках

Ответ

Операция успешно проведена:
{"Code":200,"Message":"operation ok","Data":{"OperationID: 11","OperationStatus: 10"},"KeyAES":"","Sign":""}

Запрос статуса операции

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":20003,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "TransactionID":"",
   "TerminalID":"",
   "OperationID": ,
   "Transaction":{
      "TransactionID":"",
      "TerminalID":"",
      "DateTime":""
   }
}
```

Данные операции запроса статуса операции

Поле	**Тип**	**Описание**
TransactionID	Строка	Уникальный идентификатор транзакции в системе партнера, статус которой нужно получить.
TerminalID	Строка	Уникальный идентификатор терминала в системе партнера, по которому проводилась транзакция статус которой нужно получить.
OperationID	Целое	Идентификатор операции в системе PayRun, статус которого нужно получить.

Запрос может быть инициирован по параметрам:

- либо OperationID
  в этом случае допустимо передавать пустые значения TransactionID и TerminalID
- либо по паре значений TransactionID и TerminalID
  в этом случае в OperationID не передается

* TransactionID в структуре Transaction -- непосредственно идентификатор транзакции запроса статуса операции

Статус операции при двухфакторном взаимодействии: в запросе следует указать TransactionID и TerminalID операции получения ссылки платежного виджета, при этом в ответе будет передан статус операции “pay” совершенной в данном виджете.

Для операции “pay” возможны следующие статусы:

- 5 - операция в процессе выполнения
- 7 - ожидание подтверждения: была рассчитана комиссия но оплата не была выполнена, либо (при “отложенном платеже”) требуется подтверждение/отмена списания.
- 10 - оплата проведена успешно
- 21 - оплата не удалась
- 22 - оплата отменена

Пример

Запрос:

{"Partner":{"PartnerToken":"f3347d14-8900-494c-a92f-03807dc001a1","OperationType":20003},
"Data":"{\"TransactionID\":\"321456\",\"TerminalID\":\"1\",\"Transaction\":{\"TransactionID\":\"321
466\",\"TerminalID\":\"1\",\"DateTime\":\"2019-02-11 13:37:17\"}}","KeyAES":"","Sign":""}

Ответ:
В строке OperationResponse содержится статус обработки запрошенной операции.

Операция была успешно проведена:
{"Code":200,"Message":"done","Data":{"OperationID”: 17,"OperationStatus”: 10,
“OperationResponse”:”{\"Code\":200,\"Message\":\"done\",\"Data\":{\"OperationID:
11\",\"OperationStatus: 10\",\"DateTime\":\"20180510
13:23:19\",\”OperationType\”:10102},\"KeyAES\":\"\",\"Sign\":\"\"}”},"KeyAES":"","Sign":""}

Операция была завершена с ошибкой:
{"Code":200,"Message":"done","Data":{"OperationID”: 17,"OperationStatus”: 10,
“OperationResponse”:”{\"Code\":200,\"Data\":{\"OperationStatus\":21},\"Sign\":\"\",\"KeyAES\":\"\"
,\"Message\":\"refund operation 144377 fail: 3 - Parameters not
matching\",\”OperationType\”:30201},\"KeyAES\":\"\",\"Sign\":\"\"}”},"KeyAES":"","Sign":""}

Запрос расчета комиссии по операции

Запрос

"**Partner**" - структура JSON содержащая токен партнера и код типа операции

```
{
   "PartnerToken":"",
   "OperationType":20004,
   "Locale":""
}
```

“**Data**" - Все необходимые данные операции помещаются в JSON структуру которая содержит следующие поля с именами:

```
{
   "OperationType":,
   "Sum":,
   "Transaction":{
      "TransactionID":"",
      "TerminalID":"",
      "DateTime":""
   }
}
```

Данные операции запроса статуса операции

Поле	**Тип**	**Описание**
OperationType	Целое	Тип операции по которой необходимо рассчитать комиссию.
Sum	Целое	Сумма операции для которой необходимо рассчитать комиссию.*

* договором определяется комиссия будет включена в сумму платежа, либо добавлена к ней

Пример

Запрос:

{"Partner":{"PartnerToken":"f3347d14-8900-494c-a92f-03807dc001a1","OperationType":20004},
"Data":"{\"OperationType\":104021,\"Sum\":10000,\"Transaction\":{\"TransactionID\":\"321466\",\
"TerminalID\":\"1\",\"DateTime\":\"2019-02-11 13:37:17\"}}","KeyAES":"","Sign":""}

Ответ:
В строке OperationResponse содержится статус обработки запрошенной операции.

Операция была успешно проведена:
{"Code":200,"Message":"done","Data":{"Fee": 175,"OperationStatus": 10},"KeyAES":"","Sign":""}

Операция была завершена с ошибкой:
{"Code":200,"Message":"done","Data":{"OperationID”: 17,"OperationStatus”:
21},"KeyAES":"","Sign":""}

Пример реализации криптографических преобразований на языке Go

```
package main

import (

    "bytes"
    "crypto"
    "crypto/aes"
    "crypto/cipher"
    "crypto/rand"
    "crypto/rsa"
    "crypto/x509"
    "encoding/base64"
    "encoding/pem"
    "errors"
    "fmt"
    "hash"
    "io"
    "io/ioutil"
    "log"
)

var (
    private *rsa.PrivateKey
    public  *rsa.PublicKey
    partner *rsa.PublicKey
    err     error

    pssHash hash.Hash
    newHash = crypto.SHA256

    text   = "some test text"
    data   string
```

keyAES string

```
    sign   string
)

func main() {
    pssHash = newHash.New()
    private, err = readPrivKeyFile()
    if err != nil {
        return
    }
    public = &private.PublicKey
    partner, err = readPartnerKey()
    if err != nil {
        return
    }
    log.Println("\n\n1 - crypt\n2 - sign\n3 - verify sign\n4 - decrypt\n9
- exit")
    var todo int
    for {
        fmt.Scanln(&todo)
        switch todo {
        case 1:
            encrypt()
        case 2:
            makeSign()
        case 3:
            signVerify()
        case 4:
            decrypt()
        case 9:
            return
        default:
            log.Println("\n\n1 - crypt\n2 - sign\n3 - verify sign\n4 -
decrypt\n9 - exit")
        }
    }

}
```

```
//BEGIN of general functions
func encrypt() {

    n := []byte(text)
    aesKey, err := generateKeyAES()
    if err != nil {
        log.Println(err.Error())
        return
    }
    log.Println("keyAES", aesKey)

    data, err = encryptAES(aesKey, n)

    var encryptedKeyAES []byte

    encryptedKeyAES, err = encryptRSA(aesKey, partner)

    if err != nil {
        log.Println(err.Error())
        return
    }
    keyAES = base64.StdEncoding.EncodeToString(encryptedKeyAES)

    log.Println("data:", data)
    log.Println("keyAES:", keyAES)
}

func makeSign() {
    log.Println("makeSign")
    in, err := base64.StdEncoding.DecodeString(keyAES)
    if err != nil {
        log.Println("decode keyAES error", err.Error())
        return
    }

    pssHash.Write(in)
    hashed := pssHash.Sum(nil)
    out, err := rsa.SignPKCS1v15(rand.Reader, private, newHash, hashed)
    if err != nil {
        log.Println("makeSign err:", err.Error())
```

```
        return
    }
    pssHash.Reset()

    sign = base64.StdEncoding.EncodeToString(out)
    log.Println("sign:", sign)
}

func signVerify() {
    in, err := base64.StdEncoding.DecodeString(keyAES)
    if err != nil {
        log.Println("decode keyAES error", err.Error())
        return
    }

    signIn, err := base64.StdEncoding.DecodeString(sign)
    if err != nil {
        log.Println("decode sign error:", err.Error())
        return
    }

    pssHash.Write(in)
    hashed := pssHash.Sum(nil)
    err = rsa.VerifyPKCS1v15(partner, newHash, hashed, signIn)
    if err != nil {
        log.Println("verify sign: FAIL!")
        pssHash.Reset()
        return
    }
    pssHash.Reset()
    log.Println("verify sign: OK!")
}

func decrypt() {
    key, err := base64.StdEncoding.DecodeString(keyAES)
    if err != nil {
        log.Println("decode keyAES error", err.Error())
        return
    }
```

```
    aesKey, err := decryptRSA(key, private)
    if err != nil {
        log.Println(err.Error())
        return
    }

    decodedData, err := decryptAES(aesKey, data)

    log.Println("decoded:", decodedData)
}

//END of general functions

//reading own private key
func readPrivKeyFile() (*rsa.PrivateKey, error) {
    n, err := ioutil.ReadFile("private.key")
    if err != nil {
        log.Println("read pubFile err:", err.Error())
        return nil, err
    }

    block, _ := pem.Decode(n)
    key, err := x509.ParsePKCS1PrivateKey(block.Bytes)
    if err != nil {
        log.Println("parse err:", err.Error())
        return nil, err
    }
    return key, nil
}

//reading partner's public key
func readPartnerKey() (*rsa.PublicKey, error) {
    var key *rsa.PublicKey
    n, err := ioutil.ReadFile("partner.key")
    if err != nil {
        log.Println("read pubFile err:", err.Error())
        return key, err
    }

    pemBlock, _ := pem.Decode(n)
```

```
    if pemBlock == nil {
        log.Println("pemBlock decode err")
        return key, fmt.Errorf("pemBlock decode err")
    }

    keyInt, err := x509.ParsePKIXPublicKey(pemBlock.Bytes)
    if err != nil {
        log.Println("ParsePKIXPublicKey err:", err.Error())
        return key, err
    }

    key = keyInt.(*rsa.PublicKey)
    return key, nil
}

//AES crypto functions
func encryptAES(key, text []byte) (string, error) {
    block, err := aes.NewCipher(key)
    if err != nil {
        log.Println(err.Error())
        return "", err
    }

    //text padding
    padedText := pad(text)

    ciphertext := make([]byte, aes.BlockSize+len(padedText))
    iv := ciphertext[:aes.BlockSize]
    _, err = io.ReadFull(rand.Reader, iv)
    if err != nil {
        log.Println(err.Error())
        return "", err
    }
    log.Println("encryptAES: IV =", iv)

    cbc := cipher.NewCBCEncrypter(block, iv)
    cbc.CryptBlocks(ciphertext[aes.BlockSize:], []byte(padedText))

    finalMsg := base64.StdEncoding.EncodeToString(ciphertext)
    return finalMsg, nil
```

```
}

func decryptAES(key []byte, text string) (string, error) {
    block, err := aes.NewCipher(key)
    if err != nil {
        log.Println(err.Error())
        return "", err
    }

    decodedMsg, err := base64.StdEncoding.DecodeString(text)
    if err != nil {
        log.Println(err.Error())
        return "", err
    }

    iv := decodedMsg[:aes.BlockSize]
    log.Println("decryptAES: IV =", iv)

    msg := decodedMsg[aes.BlockSize:]

    cbc := cipher.NewCBCDecrypter(block, iv)
    cbc.CryptBlocks(msg, msg)

    log.Println("padedText", string(msg))

    //text unpadding
    unpadedText, err := unpad(msg)
    if err != nil {
        log.Println(err.Error())
        return "", err
    }

    return string(unpadedText), nil
}

//AES utility functions
func generateKeyAES() ([]byte, error) {
    key := make([]byte, 16)
    _, err := rand.Read(key)
    if err != nil {
```

```
        log.Println(err.Error())
        return nil, err
    }
    return key, err
}

func pad(src []byte) []byte {
    padding := aes.BlockSize - len(src)%aes.BlockSize
    padtext := bytes.Repeat([]byte{byte(padding)}, padding)
    return append(src, padtext...)
}

func unpad(src []byte) ([]byte, error) {
    length := len(src)
    unpadding := int(src[length-1])

    if unpadding > length {
        return nil, errors.New("unpad error. This could happen when
incorrect encryption key is used")
    }

    return src[:(length - unpadding)], nil
}

//RSA crypto functions
func encryptRSA(in []byte, k *rsa.PublicKey) ([]byte, error) {
    out, err := rsa.EncryptPKCS1v15(rand.Reader, k, in)
    if err != nil {
        log.Println(err.Error())
        return out, err
    }
    return out, nil
}

func decryptRSA(in []byte, k *rsa.PrivateKey) ([]byte, error) {
    out, err := rsa.DecryptPKCS1v15(rand.Reader, k, in)
    if err != nil {
        log.Println(err.Error())
        return out, err
    }
```

```
    return out, nil
}
```

Примеры реализации партнерской интеграции

PHP

https://github.com/paycoreio/xpayua

Python

from Crypto.Hash import SHA256

from Crypto.PublicKey import RSA

from Crypto.Signature import PKCS1_v1_5

data = '''{

"signature": {

"partnerId": "partner_uuid",

"sign": ""

},

"code": “ Json строка данных запроса в формате Base64”

}

Данные которые помещаются строкой в тег code в формате Base64:

{

"request": {

"action": "request_action",

"date": "yyyy.mm.dd hh24:mi:ss",

"requestId": "request_id",

"serviceId": "service_id"

},

"data": {

}

}'''

hash = SHA256.new()

hash.update(data.encode())

```
imported_private_key = RSA.import_key(private_key)
signer = PKCS1_v1_5.new(imported_private_key)
sign = signer.sign(hash)
signed_key = b64encode(sign)
```