V1

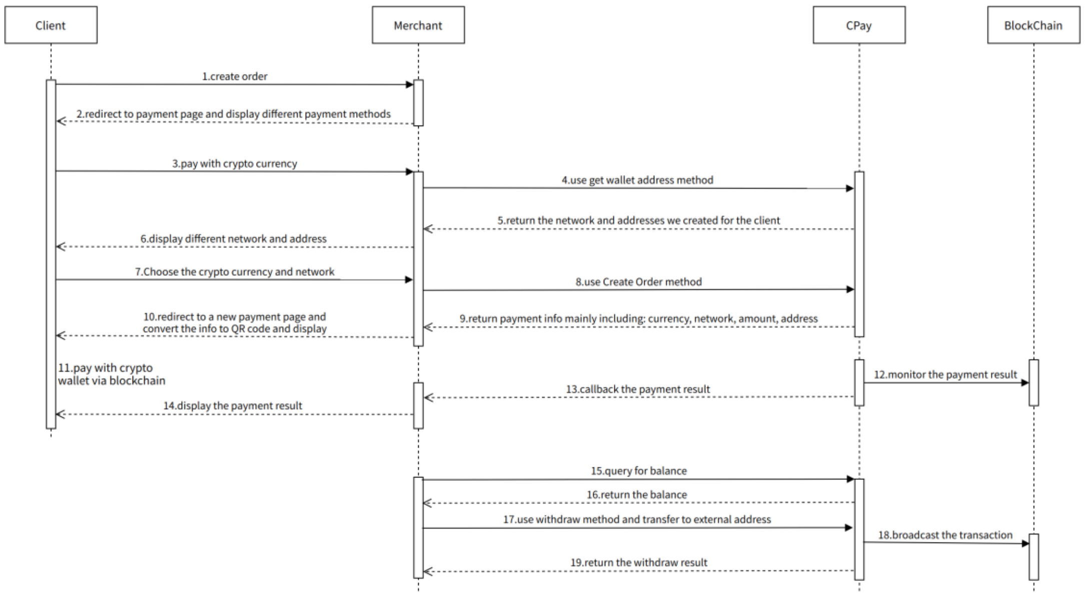
Version update instructions

Version	**update time**	**Update Content**
v1.1	29/07/2022	• Add request parameter transfer format • Add a description of the limit for the purchase of cryptocurrency • Add api signature go demo
v1.2	19/08/2022	Update Create order response message
v1.3	30/08/2022	Add webhooks
v1.4	07/09/2022	Add interface :Query Order Detail By Transaction Hash
v1.5	08/09/2022	Add interface:Query Order Detail Adjust document order
v1.6	23/09/2022	Add Credit Card Acquiring
v1.7	08/11/2022	Add success and fail url for Create Order By Credit Card
v1.8	30/01/2023	Add H2H mode,including KYC and create order V2
v1.9	03/03/2023	H2H endpoint add response field 'orderId'
v2.0	06/03/2023	Added legal currency payment interface

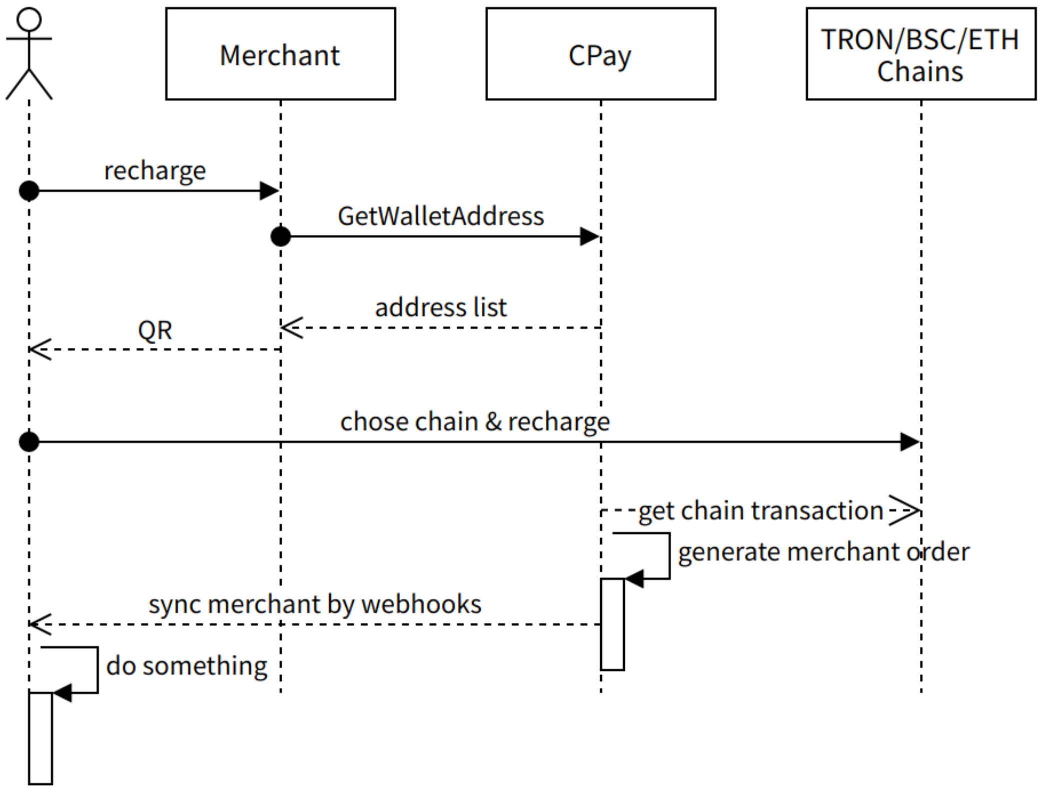
C-Pay
Interface Description

1. **Flowchart**

UML of our main service: payment and settlement based on crypto currency.


Client
Merchant
CPay
BlockChain
1.create order
2.redirect to payment page and display different payment methods
3.pay with crypto currency
4.use get wallet address method
5.return the network and addresses we created for the client
6.display different network and address
7.Choose the crypto currency and network
8.use Create Order method
10.redirect to a new payment page and convert the info to QR code and display
9.return payment info mainly including: currency, network, amount, address
11.pay with crypto wallet via blockchain
12.monitor the payment result
13.callback the payment result
14.display the payment result
15.query for balance
16.return the balance
17.use withdraw method and transfer to external address
18.broadcast the transaction
19.return the withdraw result

Cryptocurrency No-Order Mode



2. Env config

domain test : https://sandbox-api.cpay.ltd

domain prod: https://api.cpay.ltd/

3. Get WalletAddress

Description: query for the deposit address of your users.

Note: If the user want
to deposit crypto directly from his external wallet, your system needs to use this method ,so we can generate a new address for the client ,otherwise he can't
deposit .

Request URL:
https://{domain}/openapi/v1/getWalletAddress

Request method: GET

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	Long	Yes	The unique ID generated by C-Pay for merchant
userId	String(64)	Yes	User's unique ID in merchant's system
sign	String(32)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request Example**

```
HTTP
curl --location --request GET
'https://api.cpay.ltd/openapi/v1/getWalletAddress?merchantId=<merc
hantId>&userId=<userId>&sign=<sign>' \
```

- **Response parameters**

Name	Type	Description
currency	String	Currency
address	String	Address
network	String	Network

- **Response example**

```
JSON
{
    "code": 0,
    "msg": "success",
    "data": [
        {
            "address": "TRMYtLHzmTeMm1FW86LZh2LGnE2D7kJnnw",
            "currency": "USDT",
            "network": "TRON(TRC-20)"
        },
        {
            "address": "TRMYtLHzmTeMm1FW86LZh2LGnE2D7kJnnw",
            "currency": "TRX",
            "network": "TRON(TRC-20)"
        },
        {
            "address":
"0xcb5e6bc3699d1a92f80d9e1de129e500a58b5221",
            "currency": "USDT",
            "network": "BSC(BEP-20)"
        },
        {
            "address":
"0xcb5e6bc3699d1a92f80d9e1de129e500a58b5221",
            "currency": "BNB",
            "network": "BSC(BEP-20)"
        },
        {
            "address":
"0x51777edc64e66e953a576d8b90310b2a80f18f67",
            "currency": "USDT",
            "network": "Ethereum(ERC-20)"
        },
        {
            "address":
"0x51777edc64e66e953a576d8b90310b2a80f18f67",
            "currency": "ETH",
            "network": "Ethereum(ERC-20)"
        },
        {
            "address": "3Dj8spQNdFUXRP1T91egJx8ZhfDc6QECsa",
            "currency": "BTC",
            "network": "Bitcoin"
        }
    ],
    "traceid": "221117074719X9921729"
}
```

4. Payout

Description: Use this method when your clients want to withdraw crypto coins to their external wallet address.

Request URL:
https://{domain}/openapi/v1/withdraw

Request method: POST

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	Long	Yes	The unique ID generated by C-Pay for merchant
userId	String(64)	Yes	User’s unique ID in merchant’s system
merchantTradeNo	String(64)	Yes	The unique order number generated by merchant
createTime	Long	Yes	The Order time (ms) generated by merchant

cryptoCurrency	String(16)	Yes	The currency user wants to withdraw. e.g. BTC, USDT, ETH
network	String(16)	Yes	Network e.g. Bitcoin, Ethereum(ERC-20), TRON(TRC-20), BSC(BEP-20)
totalAmount	String(128)	Yes	The amount user wants to withdraw. Support 8 decimals, e.g. 66.12345678
receivedAmount	String(128)	Yes	The actual amount received by the user
toAddress	String(128)	Yes	The address user fills in to receive the withdraw
callBackURL	String(256)	No	This callBackURL will be called back after the order succeeds or fails, and the callBackURL configured in the order will be used preferentially when calling back. If it is not in the order, the callBackURL configured in the merchant platform will be used for callback
sign	String(32)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request Example**

```
HTTP
http://8.142.157.45:9075/openapi/v1/withdraw?merchantId=20000092&FiatCurrency=USD&cryptoCurrency=USDT&purchaseType=1&amount=100&sign=1d102274f2e98ad0bbfdf97c42110a748105e96182a2350a39f2d998443dabfb&merchantTradeNo=100000230&createTime=1653669353000&userId=2345
```

- **Response parameters**

Name	Type	Mandatory	Description

merchantId	String		The unique ID generated by C-Pay for merchant
cryptoCurrency	String		The currency user wants to withdraw. e.g. BTC, USDT, ETH
orderAmount	String		The amount user wants to withdraw. Support 8 decimals, e.g. 66.12345678
receivedAmount	String		The actual amount received by the user
fee	String		CPay charges a handling fee
merchantTradeNo	String		The unique order number generated by merchant
remark	String		
extInfo	String		
status	String		0:PENDING 11： PROCESSING 14:COMPLETED 15:CLOSED
orderId	String		CPay orderId
network	String		Network e.g. Bitcoin, Ethereum(ERC-20), TRON(TRC-20), BSC(BEP-20)
merchantUserId	String		User’s unique ID in merchant’s system
createTime	String		

- **Response Example**

```
JSON
{
    "code": 0,
    "msg": "ok",
    "data": {
        "fee": "0.0123"
    }
}
```

5. Fiat Currency Payout

Description: Use this method when your clients want to withdraw fiat currency to their account.

Request URL: https://{domain}/openapi/v1/openapi/v1/createFiatPayoutOrder

Request method: POST

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	Long	Yes	The unique ID generated by C-Pay for merchant
userId	String(64)	Yes	User’s unique ID in merchant’s system
merchantTradeNo	String(64)	Yes	The unique order number generated by merchant
payoutChannel	String(1)	Yes	payoutChannel 1、Pix If payoutChannel = 1, accountType and toAccount must be filled，currency must be BRL , country must be BRA 2、Bank card
amount	String(12)	Yes	The actual amount received by the user
currency	String(12)	Yes	ISO If payoutChannel = 1, country must be BRL

accountType	String(1)	Yes	If payoutChannel = 1： 1 = cpf/cnpj 2 = email 3 = mobile in format +55+AreaCode+Number as example +5500000000000 4 = random key
toAccount	String（12）	Yes	toAccount should be filled with the value of the account information selected by accountType corresponding to the payout method selected by payoutChannel If payoutChannel = 1： If accountType=1: toAccount=user pf/cnpj If accountType=2: toAccount=user email
accountCode			
country	String(12)	Yes	ISO If payoutChannel = 1, country must be BRA
createTime	Long	Yes	The Order time (ms) generated by merchant
callBackURL	String(256)	No	This callBackURL will be called back after the order succeeds or fails, and the callBackURL configured in the order will be used preferentially when calling back. If it is not in the order, the callBackURL configured in the merchant platform will be used for callback
successURL	String(256)	No	Return to the merchant address after successful payment
failURL	String(25	No	Return to merchant address after

	6)		payment failure
remark	String(256)	NO	remark
extInfo	string(256)	NO	The extension field, in json format, will be compatible with the new payout method in the future, and the special format field will be filled in here
sign	String(256)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request Example**

```
HTTP
https://sandbox-
api.cpay.ltd/openapi/v1/createFiatPayoutOrder?merchantId=20003xxx&
merchantTradeNo=test07&createTime=1663674686509&userId=1001&curren
cy=BRL&payoutChannel=1&amount=5&country=BRA&sign=b4c29ae2af4095215
2d01e08edd2652b94f035e0989d40439fc496769d643dc4&accountType=1&toAc
count=4864330xxx&callBackURL=https://www.baidu.com&successURL=http
s://cashier.cpay.ltd/succed&failURL=https://cashier.cpay.ltd/lose&
remark=remark&extInfo=
```

- **Response parameters**

Name	Type	Mandatory	Description
merchantId	String		The unique ID generated by C-Pay for merchant
merchantUserId	String		User’s unique ID in merchant’s system
merchantTradeNo	String		The unique order number generated by merchant

currency	String		The currency user wants to withdraw. e.g. BTC, USDT, ETH
orderAmount	String		The total amount of this order, including the actual payout amount + fee amount
receivedAmount	String		The actual amount received for this order
status	String		0:PENDING 11： PROCESSING 14:COMPLETED 15:CLOSED
orderId	String		CPay orderId
fee	String		CPay charges a handling fee
remark	String		
createTime	String		Timestamp in milliseconds when this order was generated

- **Response Example**

JSON
```json
{
    "code": 0,
    "msg": "success",
    "data": {
        "merchantId": "20003xxx",
        "currency": "BRL",
        "orderAmount": "5.5",
        "receivedAmount": "5",
        "fee": "0.5",
        "merchantTradeNo": "test07",
        "remark": "remark",
        "status": "11",
        "orderId": "23031008544224390xxx",
        "merchantUserId": "1001",
        "createTime": "1678438482000"
    },
    "traceid": "230310085442X2425795"
}
```

6. Query Merchant Balance

Description:

Request URL: https://{domain}/openapi/v1/getMerchantBalance

Request method: GET

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	String(64)	Yes	The unique ID generated by C.Pay for merchant
sign	String(256)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request example**

```
HTTP
curl --location --request GET
'https://api.cpay.ltd/openapi/v1/getMerchantBalance?merchantId=<merchantId>&sign=<sign>' \
```

- **Response parameters**

Name	Type	Mandatory	Description
availableBalance	String(64)		
freezeBalance	String(64)		
cryptoCurrency	String(128)		

- **Response example**

```
JSON
{
    "code": 0,
    "msg": "success",
    "data": [
        {
            "availableBalance": "0",
            "freezeBalance": "0",
            "cryptoCurrency": "ETH"
        },
        {
            "availableBalance": "0",
            "freezeBalance": "0",
            "cryptoCurrency": "TRX"
        },
        {
            "availableBalance": "0",
            "freezeBalance": "0",
            "cryptoCurrency": "EUR"
        },
        {
            "availableBalance": "0",
            "freezeBalance": "12.237",
            "cryptoCurrency": "USD"
        },
        {
            "availableBalance": "0",
            "freezeBalance": "0",
            "cryptoCurrency": "BTC"
        },
        {
            "availableBalance": "11.008",
            "freezeBalance": "0",
            "cryptoCurrency": "USDT"
        },
        {
            "availableBalance": "0",
            "freezeBalance": "0",
            "cryptoCurrency": "BNB"
        }
    ],
    "traceid": "221117075237X7604753"
}
```

7. Query ExchangeRate

Description: query the exchange rate for "Fiat to crypto" and "crypto to crypto"

Request URL: https://{domain}/openapi/v1/getExchangeRate

Request method: GET

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	Long	Yes	The unique ID generated by C-Pay for merchant
sourceCurrency	String(16)	Yes	The currency user has
targetCurrency	String(16)	Yes	The currency user wants to get
purchaseType	int	Yes	0:by sourceCurrency amount 1:by targetCurrency amount
amount	String(128)	Yes	If purchaseType=0, it means “I want to spend XXX to buy crypto ”; If purchaseType=1,it means “I want to get XXX crypto coin”
sign	String(32)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request Example**

```
HTTP
```

```
curl --location --request GET
'https://api.cpay.ltd/openapi/v1/getExchangeRate?merchantId=<merch
antId>&sourceCurrency=<sourceCurrency>&targetCurrency=<targetCurre
ncy>&purchaseType=<purchaseType>&amount=<amount>&sign=<sign>' \
```

- **Response parameters**

Name	Type	Description
exchangeRate	String(128)	Real time exchange rate

- **Response Example**

```
JSON
{
    "code": 0,
    "msg": "success",
    "data": {
        "exchangeRate": 0
    },
    "traceid": "221117075732X2324202"
}
```

Note： sourceCurrency:targetCurrency=1:exchangeRate

e.g. :sourceCurrency is USDT, targetCurrency is BTC, exchangeRate=40000.000000

8. Query ExchangeRate For Credit Card

Description: query the exchange rate for "Fiat to crypto" and "crypto to crypto"

Request URL: https://{domain}/openapi/v1/getExchangeRateForCreditCard

Request method: GET

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	Long	Yes	The unique ID generated by C-Pay for merchant
sourceCurrency	String(16)	Yes	The currency user has
targetCurrency	String(16)	Yes	The currency user wants to get
purchaseType	int	Yes	0:by sourceCurrency amount 1:by targetCurrency amount
amount	String(128)	Yes	If purchaseType=0, it means “I want to spend XXX to buy crypto ”; If purchaseType=1,it means “I want to get XXX crypto coin”
sign	String(32)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request Example**

```
HTTP
curl --location --request GET
'https://api.cpay.ltd/openapi/v1/getExchangeRateForCreditCard?merc
hantId=<merchantId>&sourceCurrency=<sourceCurrency>&targetCurrency
=<targetCurrency>&purchaseType=<purchaseType>&amount=<amount>&sign
=<sign>' \
```

- **Response parameters**

Name	Type	Description
exchangeRate	String(12	Real time exchange rate

	8)	
amountExchanged	String(128)	If the purchaseType=0, it means “when I spend Fiat, XXX crypto I will get ” If the purchaseType=1, it means “if I want to get crypto ,how much Fiat I need to pay”

- **Response Example**

```
JSON
{
    "code": 0,
    "msg": "ok",
    "data": {
        "exchangeRate": "2999.342",
        "amountExchanged": "0.03334064604"
    }
}
```

Note： sourceCurrency:targetCurrency=1:exchangeRate

e.g. :sourceCurrency is USDT, targetCurrency is BTC, exchangeRate=40000.000000

9. Query Order Detail

Description: In order to prevent merchant users from directly recharging to the C.Pay wallet address, the merchant fails to be recalled. Merchants can inquire the status of this transaction in C.Pay through the on-chain transaction HASH through this interface.

Request URL: https://{domain}/openapi/v1/getOrderDetail

Request method: GET

- **Request parameters**

Name	Type	Mandator	Description

		y	
merchantId	Long	Yes	The unique ID generated by C-Pay for merchant
merchantTradeNo	String(64)	optional	
cpayOrderId	String(64)	optional	
hash	String(128)	optional	Transaction Hash
sign	String(32)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request Example**

```
HTTP
curl --location --request GET
'http://8.142.157.45:9075/openapi/v1/getOrderDetail?merchantId=<merchantId>&merchantTradeNo=<merchantTradeNo>&hash=<hash>&cpayOrderId=<cpayOrderId>&sign=<sign>' \
```

- **Response parameters**

Name	Type	Description
orderId	String	CPay orderId
status	String	0:PENDING 11： PROCESSING 14:COMPLETED 15:CLOSED
merchantTradeNo	String	

merchantUserId	String	
hash	String	
actualAmount	String	C.Pay received
receivedAmount	String	Merchant account received
pledgeAmount	String	
fee	String	C.Pay charges a handling fee
currency	String	
createTime	String	
merchantId	String	
extInfo	String	
network	String	
remark	String	

- **Response Example**

```
JSON
{
    "code": 0,
    "msg": "success",
    "data": {
        "actualAmount": "10",
        "createTime": 1663155010000,
        "currency": "USDT",
        "extInfo": "{\"reason\":\"Transaction cannot be
completed\",\"sceneTag\":\"EcAcquiring\"}",
        "fee": "0",
        "hash":
"c4dbbf5737c52072817dc92fa09c41386f8024fee883270b92bfc1943616f957"
,
        "merchantId": 20003092,
        "merchantTradeNo": "7b785240e426694635bb09fb101ae241",
        "merchantUserId": "test002",
        "network": "TRON",
        "orderId": "2209141130105863014",
        "pledgeAmount": "0",
        "receivedAmount": "10",
        "remark": "充值单补单",
        "status": "14"
    },
    "traceid": "221117082056X6478814"
}
```

10. Exchange

Description: query for exchange between two crypto coins

Request URL:
https://{domain}/openapi/v1/exchange

Request method: POST

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	Long	Yes	The unique ID generated by C-Pay for merchant
userId	String(64)	Yes	User’s unique ID in merchant’s system
merchantTradeNo	String(64)	Yes	The unique order number generated by merchant
createTime	Long	Yes	The Order time (ms) generated by merchant
sourceCurren	String(16	Yes	The currency user has

cy	)		
sourceAmount	String(128)	Yes	Amount of source currency
targetCurrency	String(16)	Yes	The currency user wants to get
sign	String(32)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request example**

```
HTTP
curl -X POST 'https://domain/openapi/v1/exchange' -d
'merchantId=200001111&userId=xxx-
1001&merchantTradeNo=100&createTime=1655899200000&sourceCurrency=U
SDT&sourceAmount=9.12345678&targetCurrency=BTC&sign=xxxxxxxxxxxxxx
xxx'
```

- **Response parameters**

Name	Type	Description
exchangeRate	String	The actual exchange rate between source currency and target currency
targetAmount	String	The actual amount of target currency the user gets after the exchange.
targetCurrency	String	The currency user wants to get
fee	String	Transaction fee amount. The fee currency is the same as the target currency.

- **Response example**

```
JSON
{
    "code": 0,
    "msg": "ok",
    "data": {
        "targetCurrency": "BTC",
        "targetAmount": "0.997",
        "exchangeRate": "40000.000000",
        "fee": "0.003"
    }
}
```

11. Submit KYC information

Description: When using the H2H mode, user KYC data must be submitted to CPAY through this interface

Request URL:
https://{domain}/openapi/v1/submitKYC

Request method: POST

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	Long	Yes	The unique ID generated by C-Pay for merchant
data		Yes	JSON format data, the sample is as follows: [{ "userId": "231xsd23", "firstName": "hart", "middleName":"", "lastName": "lee", "brithday": "1987-05-01", "gender": "male/female",

			"country": "RU", "iddCode": "7", "personalCode": "12341234123", "certificateDateStart": "2008-01-01", "certificateDateEnd": "2038-01-01", "IdCardUrls": ["http(s)://****/pi.jpg",], "documentUrls": ["http(s)://****/pi.pdf",], "email": "asdf@gmail.com", "mobile": "23452345" } ,] note:Up to 1000 pieces of data at a time
sign	String(32)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request example**

```
HTTP
curl --location --request POST
'https://api.cpay.ltd/openapi/v1/submitKYC
```

- **Response parameters**

Name	Type	Mandatory	Description

code	String		
msg	String		
data	Object		
traceid	String		
count	Long		quantity accepted

- **Response example**

```
JSON
{
    "code": 0,
    "msg": "success",
    "data": {
        "count": 10
    },
    "traceid": "221117082830X664937"
}
```

12. Payment without Payment Page for Credit card

Description: H2H mode payment

Request URL:
https://{domain}/openapi/v1/pwpp

Request method: POST

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	String(64)	Yes	The unique ID will be provided by C.Pay for merchant
merchantTradeNo	String(64)	Yes	The unique order number generated by merchant
lang	String	No	Default English
userId	String	Yes	User’s unique ID in merchant’s system
storeCc	String	No	Specifies, if the credit card should be stored in case of success: 0 = Do not store the card (default) 1 = Store the card In order to store the credit card, Credit Card Storage service should be enabled in your account.
ccStorageID	String	No	When charging stored credit card, the ID of the stored card.
firstName	String	Yes	First name on credit card
lastName	String	Yes	Last name on credit card
email	String	Yes	
iddCode	String	Yes	https://en.wikipedia.org/wiki/List_of_country_calling_codes Example: 35
mobile	String	Yes	Example: 712345678
addressLine	String	Yes	Alphanumeric, _ , -, . (1-50)

country	String	Yes	ISO3166-1 2-digit country code
city	String	Yes	Alphabetic (1-15)
zip	String	Yes	Billing zip code
cardNumber	String	Yes	Bank Card Number
cvv	String	Yes	CVV code
expDate	String	Yes	Card expired Month / Card expired Year Example: *12/27*
ip	String	Yes	IPV4
currency	String(64)	Yes	ISO4217（only USD, EUR supported）
amount	String(128)	Yes	The precision is two decimal places,higher precision rounds down. Example: 19.011 -> 19.01 1.528 -> 1.52
products	String(16)	Yes	Merchant side order commodity JSON array. example： "[{"name":"iphone 11","price":"5300.00","num":"2","currency":"CNY"},{"name":"macBook","price":"1234.00","num":"1","currency":"USD"}]"
payChannel	String(16)	No	Payment channel example： payChannel1 or payChannel2

clientHTTPAccept	String(256)	Yes	**request header that allows a characteristic string** **example：** **"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7"**
clientHTTPUserAgent	String(256)	Yes	**ClientHTTPUserAgent** **example：** **" Mozilla/5.0 (iPhone; CPU iPhone OS 11_3_1 like Mac OS X) AppleWebKit/603.1.30 (KHTML, like Gecko) Version/10.0 Mobile/14E304 Safari/602.1"**
callBackURL	String(256)	No	This callBackURL will be called back after the order succeeds or fails, and the callBackURL configured in the order will be used preferentially when calling back. If it is not in the order, the callBackURL configured in the merchant platform will be used for callback
successURL	String(256)	No	Return to the merchant address after successful payment
failURL	String(256)	No	Return to merchant address after payment failure
createTime	Long	Yes	The Order time (ms) generated by merchant
sign	String(256)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request example**

```
HTTP
curl --location --request POST
'https://api.cpay.ltd/openapi/v1/pwpp?merchantId=<merchantId>&fiat
Currency=<fiatCurrency>&cryptoCurrency=<cryptoCurrency>&purchaseTy
pe=<purchaseType>&amount=<amount>&sign=<sign>&merchantTradeNo=<mer
chantTradeNo>&createTime=<createTime>&userId=<userId>' \
```

- **Response parameters**

Name	Type	Mandatory	Description
code	String		
msg	String		
data	Object		
traceid	String		
orderId	string		This payment order id. After the jump, `orderId` will be spliced to `successURL` or `failURL` as a URL parameter: `<Your_SuccessURL>?txnRef=<orderId>`
pageData	string		HTML original document, jump to 3DS or successURL or failURL. You need open it or render it on your page.
redirectURL	string		The url that needs to be redirected

ccStorageID	string		When charging stored credit card, the ID of the stored card.

- **Response example**

```
JSON
{
    "code": 0,
    "msg": "success",
    "data": {
        "orderId": "2303031006533582O9014",
        "pageData": "\<\!DOCTYPE html\>\<html
lang=\"en\">...<\/html\>",

       "ccStorageID":"xxxxxxxxxxxxxxxxx"
    },
    "traceid": "221117082830X664937"
}
```

```
JSON
{
    "code": 90751015,
    "msg": "pay fail",
    "data": {
        "orderId": "",
        "pageData": "",

        "ccStorageID":""
    },
    "traceid": "221117082830X664937"
}
```

13. Create Order

Description:

Request URL: https://{domain}/openapi/v1/createOrder

Request method: POST

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	String(64)	Yes	The unique ID generated by C.Pay for merchant
merchantTradeNo	String(64)	Yes	The unique order number generated by merchant
createTime	Long	Yes	The Order time (ms) generated by merchant
userId	String(64)	Yes	User’s unique ID in merchant’s system
amount	String(128)	Yes	
cryptoCurrency	String(16)	Yes	USDT、ETH、BTC、BNB
callBackURL	String(256)	No	This callBackURL will be called back after the order succeeds or fails, and the callBackURL configured in the order will be used preferentially when calling back. If it is not in the order, the callBackURL configured in the merchant platform will be used for callback
successURL	String(256)	No	Return to the merchant address after successful payment
failURL	String(256)	No	Return to merchant address after payment failure
sign	String(256)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request example**

```
C++
curl --location --request POST
'https://api.cpay.ltd/openapi/v1/createOrder?merchantId=<merchantI
d>&merchantTradeNo=<merchantTradeNo>&userId=<userId>&cryptoCurrenc
y=<cryptoCurrency>&amount=<amount>&createTime=<createTime>&sign=<s
ign>&callBackURL=<callBackURL>&successURL=<successURL>&failURL=<fa
ilURL>' \
```

- **Response parameters**

Name	Type	Mandatory	Description
merchantId	String(64)		CPay orderId
cryptoCurrency	String(64)		
orderAmount			
receivedAmount			
pledgeAmount			
merchantTradeNo			
cashierURL			
remark			
extInfo			
status			0:PENDING 11： PROCESSING 14:COMPLETED means success 15:CLOSED means rejeected or errored

orderId			
network			
merchantUserId			
returnURL			
fee			
createTime			

- **Response example**

```
JSON
 "code": 0,
    "msg": "success",
    "data": {
        "orderId": "221116112637772768014",
        "merchantId": "20003092",
        "cryptoCurrency": "USDT",
        "orderAmount": "3",
        "receivedAmount": "3",
        "merchantTradeNo": "test114",
        "cashierURL":
"https://cashier.cpay.ltd/payment?orderId=221116112637772768014&sign=639ae2e78bd3456afab6dfe607c56848de50f3f755cdfeb5cda007626bbeabff",
        "returnURL": "",
        "successURL": "https://cashier.cpay.ltd/succed",
        "failURL": "https://cashier.cpay.ltd/lose",
        "remark": "",
        "extInfo": ""
    },
    "traceid": "221117083310X220052"
}
```

14. Create Order By Credit Card

Description:

Request URL: https://{domain}/openapi/v1/createOrderByCreditCard

Request method: POST

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	String(64)	Yes	The unique ID generated by C.Pay for merchant
merchantTradeNo	String(64)	Yes	The unique order number generated by merchant
userId	String	Yes	User’s unique ID in merchant’s system
currency	String(64)	Yes	ISO 4217（Currently only USD EUR BRL is supported）
amount	String(128)	Yes	
products	String(16)	Yes	Merchant side order commodity JSON array. example： "[,{"name":"macBook","price":"1234.00","num":"1","currency":"USD"}]"
country	String	Yes	ISO3166-1
email	String	Yes	
ip		Yes	IPV4/V6
lanuage	String	No	The default language for the UI text in the window. If omitted, language is taken from user's

			browser settings.
			
			Available values are:
			en-US = English (United States)
			ru-RU = русский (Россия)

callBackURL	String(256)	No	This callBackURL will be called back after the order succeeds or fails, and the callBackURL configured in the order will be used preferentially when calling back. If it is not in the order, the callBackURL configured in the merchant platform will be used for callback
successURL	String(256)	No	Return to the merchant address after successful payment
failURL	String(256)	No	Return to merchant address after payment failure
createTime	Long	Yes	The Order time (ms) generated by merchant
sign	String(256)	Yes	See API Signature

Note: This interface uses the form form to receive parameters

- **Request example**

```
HTTP
curl --location --request POST
'https://api.cpay.ltd/openapi/v1/createOrderByCreditCard?merchantI
d=<merchantId>&merchantTradeNo=<merchantTradeNo>&userId=<userId>&c
urrency=<currency>&amount=<amount>&products=<products>&callBackURL
=<callBackURL>&createTime=<createTime>&sign=<sign>&country=<countr
y>&email=<email>&ip=<ip>&failURL=<failURL>&successURL=<successURL>
' \
```

- **Response parameters**

Name	Type	Mandatory	Description
merchantId	String(64)		CPay orderId
merchantTradeNo	String(64)		
orderId	String(64)		
orderStatus	String(2)		0:PENDING 11：PROCESSING 14:COMPLETED means success 15:CLOSED means rejeected or errored
orderCurrency	String(16)		
orderAmount	String(16)		
receivedAmount	String(16)		
pledgeAmount	String(16)		
fee	String(16)		
cashierURL	String(256)		
successURL	String(256)	No	Return to the merchant address after successful payment
failURL	String(256)	No	Return to merchant address after payment failure
sign	String(		

	256)		

- **Response example**

```
JSON
{
    "code": 0,
    "msg": "success",
    "data": {
        "merchantId": 20003092,
        "merchantTradeNo": "test901",
        "orderId": "22110810121223831OO14",
        "orderStatus": "15",
        "orderCurrency": "USD",
        "orderAmount": "2",
        "receivedAmount": "0.00",
        "pledgeAmount": "",
        "fee": "0.00",
        "cashierURL":
"https://gw2.mcpayment.net/api/v6/paymentPage/B462EEE0EBF6BAEB7D11
38877435F8ACDFDB6333",
        "successURL": "",
        "failURL": "",
        "sign":
"e344e3a24ea4163c5b86482165e41d6ecad8db874750ec34ac474b78a153a933"
    },
    "traceid": "221117083632X2666756"
}
```

15. WebHooks(Callback URL)

C.Pay sends notifications through webhooks to inform your system about events that occur in the balance platform. These events include when there are incoming funds or a payout was initiated.

When an event occurs, C.Pay makes an HTTP POST request to a URL on your server and includes the details of the event in the request body.

You can use notifications to build your implementations. For example, you can use the information to update balances in your own dashboards or to keep track of incoming funds.

Request URL: https://{merchant_domain}/{merchant_uri}

Request method: POST

- **Request parameters**

Name	Type	Mandatory	Description
merchantId	String(64)	Yes	The unique ID generated by C.Pay for merchant
merchantTradeNo	String(256)	No	
merchantUserId	String	No	
orderId	String	No	
orderStatus	String	Yes	0:PENDING 11： PROCESSING 14:COMPLETED 15:CLOSED
hash	String	No	
actualAmount	String	Yes	C.Pay received
receivedAmount	String	Yes	Merchant account received
fee	String	Yes	C.Pay charges a handling fee
cryptoCurrency	String	No	
network	String	No	
remark	String	No	
extInfo	String	No	

createTime	String	Yes	
sign	String	No	Provider by C.Pay for merchant

- **Response parameters**

webhook server returns OK or FAIL text

16. API Signature

For all methods in the api document, empty parameters should not be involved in the signature.

0、API Example:

```
Plain Text
curl
https://domain/openapi/v1/getSth?xx=1001&yy=&aa=hello&sign=Vs23424
SHW

curl -X POST 'https://domain/openapi/v1/updateSth' -d
'xx=1001&yy=&aa=hello&sign=Vs23424SHW'
```

1、All parameters will be sorted by parameter name and stitched into a character string, e.g.: `aa=hello&xx=1001`

2、Add the secret key we provide at the end of the string: `aa=hello&xx=1001`, you will get: `aa=hello&xx=1001&key=aaaaaaaaaaxxxxxx`

- *secret key will be given through email*
- *Given the value of the yy parameter is an empty string, therefore it is ignored here.*

3、Encrypt HmacSha256 the string generated in step 2, and convert it to lowercase. Then you obtain a value which is the sign parameter of the API.

sign=HmacSha256(`aa=hello&xx=1001&key=aaaaaaaaaaxxxxxx`)

sign=sign.ToLower()

Response Code

Code	Description
0	ok
1	fail

Java demo

```
Java

package com.lianjing.controller;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.util.*;

public class CheckSign {
    public static void main(String[] args) {
        String key = "*iu6hufsi%^54fyt";
        Map<String, Object> params = new HashMap<>();
        params.put("amount","32");
        params.put("merchantTradeNo","10181633121136403722 32");
        params.put("merchantId","20000092");
        params.put("createTime","20220726100218");
        params.put("fiatCurrency","USD");
        params.put("userId","10181633");
        params.put("cryptoCurrency","USDT");
        params.put("purchaseType","1");
        Collection keySet = params.keySet();
        List list = new ArrayList(keySet);
        Collections.sort(list);
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < list.size(); i++) {
            sb.append(list.get(i) + "=" +
params.get(list.get(i))+"&");
        }
        sb.append("key="+key);
        System.out.println(sb.toString());
        try {
            System.out.println(HMACSHA256(sb.toString(),key));
        } catch (Exception e) {
            throw new RuntimeException(e);
        }

    }
    public static String HMACSHA256(String data, String key)
throws Exception {

        Mac sha256_HMAC = Mac.getInstance("HmacSHA256");

        SecretKeySpec secret_key = new
SecretKeySpec(key.getBytes("UTF-8"), "HmacSHA256");

        sha256_HMAC.init(secret_key);

        byte[] array = sha256_HMAC.doFinal(data.getBytes("UTF-
8"));

        StringBuilder sb = new StringBuilder();

        for (byte item : array) {

            sb.append(Integer.toHexString((item & 0xFF) |
0x100).substring(1, 3));

        }

        return sb.toString().toLowerCase();

    }
}
```

Go demo

```
Go
func CheckSign(p interface{}) error {

   bs, err := json.Marshal(p)
   if err != nil {
      return err
   }
   m := make(map[string]interface{})
   err = json.Unmarshal(bs, &m)
   if err != nil {
      return err
   }

   mid, ok := m["merchantId"]
   if !ok || len(mid.(string)) == 0 {
      return errors.New("invalid param merchantId")
   }
   sig, ok := m["sign"]
   if !ok || len(sig.(string)) == 0 {
      return errors.New("invalid param sign")
   }

   key := PayConf.Key
   if sig.(string) == genSign(m, key) {
      return nil
   }
   return errors.New("verify sign fail")
}
```

PHP Demo

```php
<?php
 $arr = [
            'amount'=>'100',
            'merchantTradeNo'=>'test0001',
            'merchantId'=>'20000387',
            'createTime'=>'1658387195',
            'fiatCurrency'=>'USD',
            'userId'=>'83453',
            'cryptoCurrency'=>'USDT',
            'purchaseType'=>'1'
        ];

ksort($arr);
$url = '';
if (is_array($arr) && count($arr)>0) {
    foreach ($arr as $k => $v) {
    $url = $url . "{$k}={$v}&";
    }
}
$url = $url.'key=z6h3cuw9grtbun9sgg5u1j35u72hycq0';
$secret = "z6h3cuw9grtbun9sgg5u1j35u72hycq0";
echo json_encode($arr);
echo hash_hmac("sha256", $url, $secret);
```