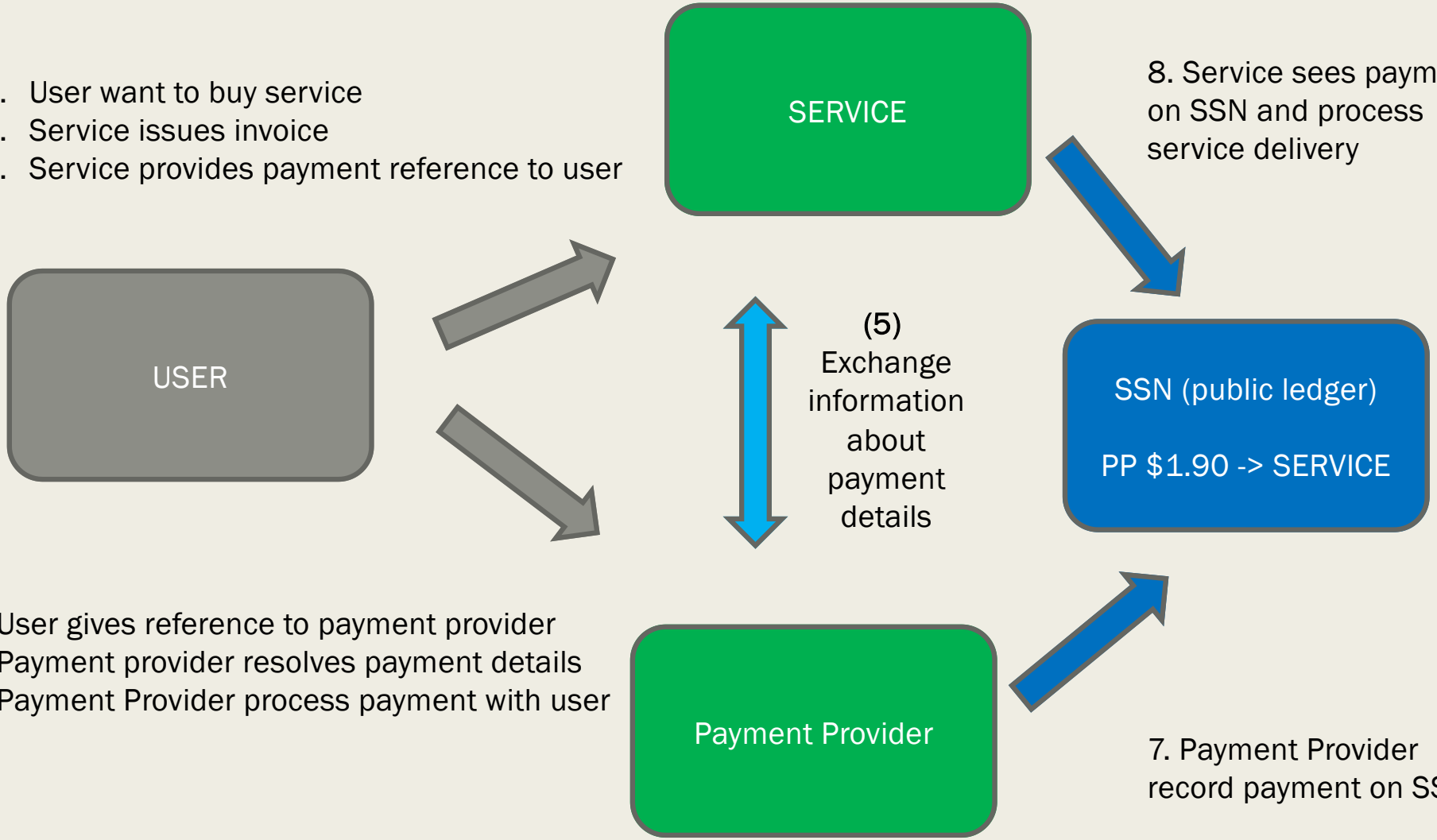
SSN MAKING PAYMENTS

Road to integration

Workflow to process payments

- SSN provides the specification for
  - *hand over of the payment details (entry point for the payment)*
  - *recording of the payment authorization (exit point for the payment)*
- Payment Provider implements
  - *Authentication of the users*
  - *Authorization of the payment*
  - *Securing of funds (move money from user to escrow account)*

1. User want to buy service
2. Service issues invoice
3. Service provides payment reference to user



8. Service sees payment on SSN and process service delivery

4. User gives reference to payment provider
5. Payment provider resolves payment details
6. Payment Provider process payment with user

7. Payment Provider record payment on SSN

Gather payment details

- Payment provider can get payment details:
  1. *From scanning QR code*
  2. *From payment address*

- Payment address can be submitted via
  - *Direct entry by user*
  - *From a API call to the payment provider by merchant*

Payment address handover

Merchant calls public API of service provider

```
https://pay.provider.com/charge/onetime/012874578*soyo.sabay.com
```

- Public API can be:
  1. *One time payment*
  2. *Recurring payment*
  3. *Pre-authorized payment*

QR code (TR003)

- SSN defines QR codes following EMV standard
- Payment Provider application needs to scan the QR code and decode it

00 02 01 = QR code version 01
01 02 12 = Dynamic Code
29 67 = Network and Account detail
    00 11 digital.ssn
    05 56 GA3336CDCBK0FJU0HZA3EYDHML7WNB5TJKA4OYF5
5204 4812 = Business Classifier (shop, restaurant e.g.)
5303 840 = currency requested = KHR
5405 80000 = amount 80,000
5802 KH = Business location (country)
5914 ABC PHONE SHOP = Business name
6010 PHNOM PENH = Business location (city)
6212 01081925-001 = memo for transaction
6304 888B

Payment address (TR002, TG004)

- Payment provider receives payment address via API or from user input
- Payment address is a text string in the format:

  invoice_id * service.provider.domain

- Payment provider looks up merchants resolver from domain
- Query address resolver to get payment details
- Payment details are encrypted and must be decrypted, see TG004 for examples

- Sabay also provides a public address resolver, which can be queried (think DNS)

  https://pa.ssn.digital/{payment_address}

payment-identifier * service.provider.domain

REQUEST:

```
{
"asset_issuer”     : "PUBLIC_KEY_OF_ASSET_ISSUING_ACCOUNT",
"public_key”       : "PUBLIC_KEY_OF_SIGNER",
"payment_address” : ”012874578*soyo.sabay.com"
}
```

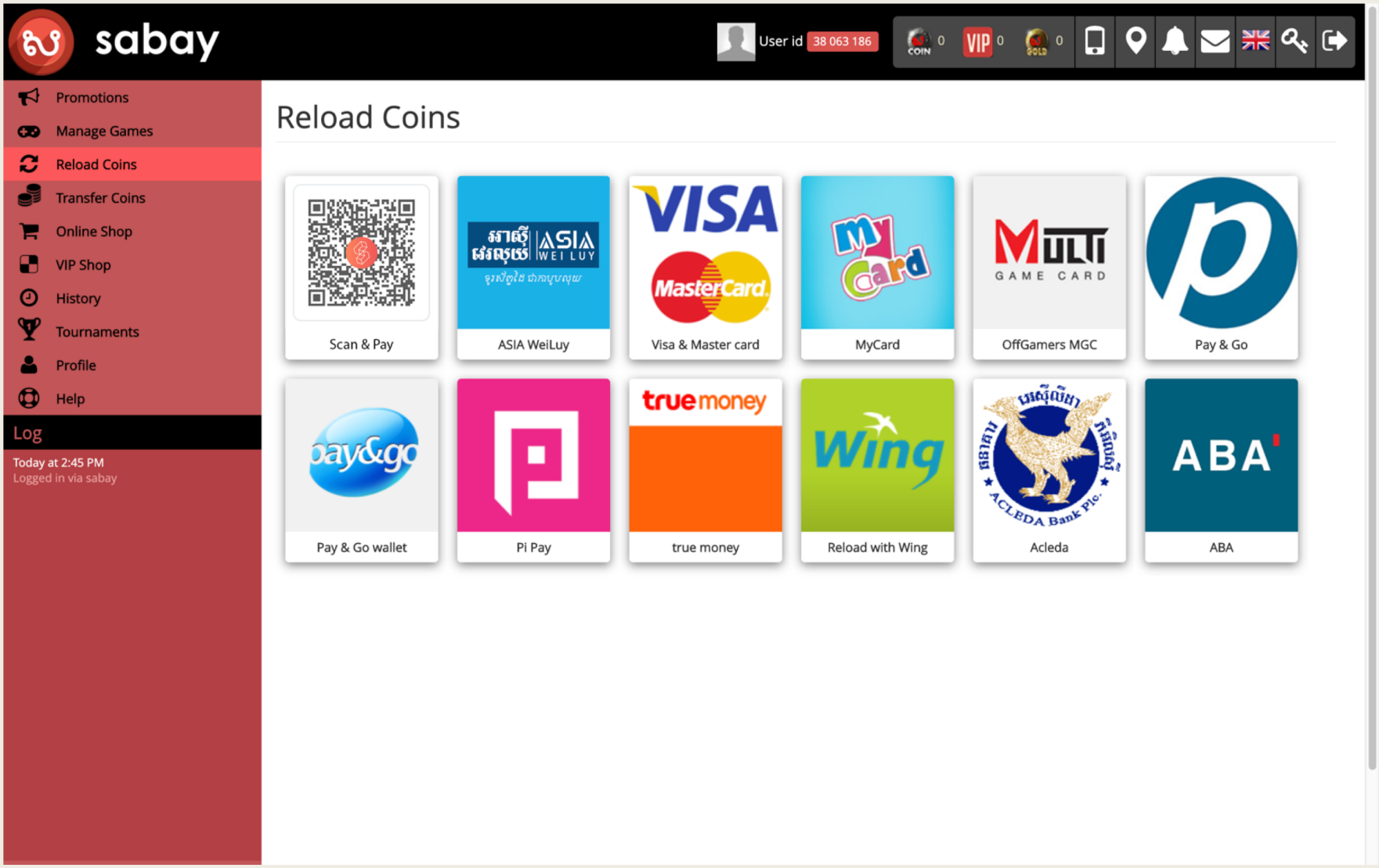
RESPONSE (after decryption):

```
{
 "network_address": "GDTXTOPAMGXDSNHAOVLGPM...PUBLIC_KEY_OF_MERCHANT",
 "public_key": "PUBLIC_KEY_OF_SIGNER",
 "asset_code": "USD",
 "payment_type": ”merchant",
 "details": {
        "memo": ”sub:012874578",
        "service_name": ”SOYO",
        "payment_info": ”Payment for monthly subscription",
        "payment": {
            "amount": 1.90,
            "asset_code": "USD”
}}}
```

QR CODE PAYMENT

No API

Payment details provided by merchant in QR code


sabay
User id 38 063 186
COIN 0
VIP 0
GOLD 0
Promotions
Manage Games
Reload Coins
Transfer Coins
Online Shop
VIP Shop
History
Tournaments
Profile
Help
Log
Today at 2:45 PM
Logged in via sabay
Reload Coins
Scan & Pay
ASIA WeiLuy
Visa & Master card
MyCard
OffGamers MGC
Pay & Go
Pay & Go wallet
Pi Pay
true money
Reload with Wing
Acleda
ABA

QR Code payment

- From your app, scan QR code with phone camera
- Decode image using EMV standard (SSN:TR003)
- If amount is missing, request user input
- All other information to make a payment are presented in the QR code
- Request user to agree to payment
- Move money from user to escrow account
- Execute payment on SSN
- Show SSN transaction ID to user

Execute on SSN in 5 lines of code

```
# python example for payment

from stellar_base.builder import Builder

# we know | defaults
SSN    = 'https://horizon.testing.ssn.digital'
SSNPW = 'ssn_testing_network'
PP_SK = 'SECRET_KEY_TO_SIGN_TRANSACTION'
PP_PK = 'PUBLIC_KEY_OF_ASSET_ISSUING_ACCOUNT'

# input
RECV_PK     = 'FROM_QR' # Merchant Account
MEMO        = 'FROM_QR'
ASSET_CODE = 'FROM_QR' # USD or KHR
AMOUNT      = 'FROM_QR_OR_ASK' # or ask user


...
```

```
...

# 1) setup transaction builder
builder = Builder(
    secret       =  PP_SK,
    network      =  SSNPW,
    horizon_uri =  SSN)

# 2) define payment operation
builder.append_payment_op(
    destination  = RECV_PK,
    amount       = AMOUNT,
    asset_code   = ASSET_CODE,
    asset_issuer = PP_PK
    )

# 3) add memo to transaction
builder.add_text_memo(MEMO)

# 4) sign
builder.sign()

# 5) submit
response = builder.submit()

# show result
notify_user(response)
```

References for QR code payments

- Documentation for QR: https://github.com/sabay-digital/org.ssn.doc.public/blob/master/tr/tr003.md
- Payment Provider documentation https://github.com/sabay-digital/org.ssn.doc.public/blob/master/tg/tg001.md
- Python Stellar SDK
- https://github.com/StellarCN/py-stellar-base

DIRECT PAYMENT

No API

Payment details hosted by Merchant

Direct payment via payment address

- Request input from user for the payment address in your application (input can also be a part of the address, for example a user ID only)
- Resolve payment address
- Request user to agree to payment
- Move money from user to escrow account
- Make transaction on SSN
- Show SSN transaction ID to user

Execute on SSN in 5 lines of code

```
# python example for payment

from stellar_base.builder import Builder

# we know | defaults
SSN   = 'https://horizon.testing.ssn.digital'
SSNPW = 'ssn_testing_network'
PP_SK = 'SECRET_KEY_TO_SIGN_TRANSACTION'
PP_PK = 'PUBLIC_KEY_OF_ASSET_ISSUING_ACCOUNT'

# input
RECV_PK    = 'FROM_PA_RESOLVER' # Merchant Acct
MEMO       = 'FROM_PA_RESOLVER'
ASSET_CODE = 'FROM_PA_RESOLVER' # USD or KHR
AMOUNT     = 'FROM_PA_RESOLVER'
...
```

```
...

# 1) setup transaction builder
builder = Builder(
    secret      =  PP_SK,
    network     =  SSNPW,
    horizon_uri =  SSN)

# 2) define payment operation
builder.append_payment_op(
    destination  = RECV_PK,
    amount       = AMOUNT,
    asset_code   = ASSET_CODE,
    asset_issuer = PP_PK
    )

# 3) add memo to transaction
builder.add_text_memo(MEMO)

# 4) sign
builder.sign()

# 5) submit
response = builder.submit()

# show result
notify_user(response)
```

References for recurring payments

- Documentation for API:
  https://api-reference.ssn.digital/?urls.primaryName=SSN%20Cashier%20API
- Payment Provider documentation
  https://github.com/sabay-digital/org.ssn.doc.public/blob/master/tg/tg001.md
- Resolve payment address
  https://api-reference.ssn.digital/?urls.primaryName=SSN%20payment%20address%20resolver%20API
- Examples to decrypt payment address resolver response
  https://github.com/sabay-digital/org.ssn.doc.public/blob/master/tg/tg004.md
- Python Stellar SDK
- https://github.com/StellarCN/py-stellar-base

ONE TIME PAYMENT

API hosted by payment provider

Payment details hosted by Merchant

One time payment API hosted by payment provider

- Build API to receive the call from the merchant
  https://pay.provider.com.kh/charge/onetime/{payment_address}
- Resolve payment address
- Authenticate user and request user to agree to payment (web, in-app, via USSD)
- Move money from user to escrow account
- Make transaction on SSN
- Show SSN transaction ID to user

Execute on SSN in 5 lines of code

```
# python example for payment

from stellar_base.builder import Builder

# we know | defaults
SSN   = 'https://horizon.testing.ssn.digital'
SSNPW = 'ssn_testing_network'
PP_SK = 'SECRET_KEY_TO_SIGN_TRANSACTION'
PP_PK = 'PUBLIC_KEY_OF_ASSET_ISSUING_ACCOUNT'

# input
RECV_PK    = 'FROM_PA_RESOLVER' # Merchant Acct
MEMO       = 'FROM_PA_RESOLVER'
ASSET_CODE = 'FROM_PA_RESOLVER' # USD or KHR
AMOUNT     = 'FROM_PA_RESOLVER'
...
```

```
...

# 1) setup transaction builder
builder = Builder(
    secret      =  PP_SK,
    network     =  SSNPW,
    horizon_uri =  SSN)

# 2) define payment operation
builder.append_payment_op(
    destination  = RECV_PK,
    amount       = AMOUNT,
    asset_code   = ASSET_CODE,
    asset_issuer = PP_PK
    )

# 3) add memo to transaction
builder.add_text_memo(MEMO)

# 4) sign
builder.sign()

# 5) submit
response = builder.submit()

# show result
notify_user(response)
```

References for recurring payments

- Documentation for API:
  https://api-reference.ssn.digital/?urls.primaryName=SSN%20Cashier%20API
- Payment Provider documentation
  https://github.com/sabay-digital/org.ssn.doc.public/blob/master/tg/tg001.md
- Resolve payment address
  https://api-reference.ssn.digital/?urls.primaryName=SSN%20payment%20address%20resolver%20API
- Examples to decrypt payment address resolver response
  https://github.com/sabay-digital/org.ssn.doc.public/blob/master/tg/tg004.md
- Python Stellar SDK
- https://github.com/StellarCN/py-stellar-base

RECURRING PAYMENT

API hosted by payment provider
Payment details hosted by Merchant
Recurring record store with Payment Provider

Recurring Payments – add subscription API hosted by payment provider

- Build API to receive the call from the merchant
  https://pay.provider.com.kh/recurring/{payment_address}

1. A URL which when called will ask the user to authenticate himself
2. Record the following in your database :
   1. *user id at at the payment provider (e.g. phone number, from authentication)*
   2. *payment address (from request)*
   3. *the recurring details, (amount, currency, period from request)*

- Call back merchant and confirm subscription
  (call back url in header send by merchant X-SSN-service-callback)

Recurring Payments – make payment

On the date when then recurring payment is due, run the following:

- Resolve payment address
- Move money from user to escrow account
- Make transaction on SSN
- Move subscription date forward for the period

Execute on SSN in 5 lines of code

```
# python example for payment

from stellar_base.builder import Builder

# we know | defaults
SSN   = 'https://horizon.testing.ssn.digital'
SSNPW = 'ssn_testing_network'
PP_SK = 'SECRET_KEY_TO_SIGN_TRANSACTION'
PP_PK = 'PUBLIC_KEY_OF_ASSET_ISSUING_ACCOUNT'

# input
RECV_PK    = 'FROM_PA_RESOLVER' # Merchant Acct
MEMO       = 'FROM_PA_RESOLVER'
ASSET_CODE = 'FROM_DATABASE_RECORD'
AMOUNT     = 'FROM_DATABASE_RECORD'
...
```

```
...

# 1) setup transaction builder
builder = Builder(
    secret      =  PP_SK,
    network     =  SSNPW,
    horizon_uri =  SSN)

# 2) define payment operation
builder.append_payment_op(
    destination  = RECV_PK,
    amount       = AMOUNT,
    asset_code   = ASSET_CODE,
    asset_issuer = PP_PK
    )

# 3) add memo to transaction
builder.add_text_memo(MEMO)

# 4) sign
builder.sign()

# 5) submit
response = builder.submit()

# show result
notify_user(response)
```

References for recurring payments

- Documentation for API:
  https://api-reference.ssn.digital/?urls.primaryName=SSN%20Cashier%20API
- Payment Provider documentation
  https://github.com/sabay-digital/org.ssn.doc.public/blob/master/tg/tg001.md
- Resolve payment address
  https://api-reference.ssn.digital/?urls.primaryName=SSN%20payment%20address%20resolver%20API
- Examples to decrypt payment address resolver response
  https://github.com/sabay-digital/org.ssn.doc.public/blob/master/tg/tg004.md
- Python Stellar SDK
- https://github.com/StellarCN/py-stellar-base

GETTING STARTED

ready

Test network

- Test network is online
- Please create your own keys
- Join telegram group and request funding to setup an account
- Write your code and execute on test network

Test network endpoint : https://horizon.testing.ssn.digtial
Test network identifier : ssn_testing_network
Lab tools for testing : https://lab.testing.ssn.digital/

High speed transaction API

- SSN provides a high speed production API for making transactions

  SSN 3rd party API https://api-reference.ssn.digital/

- The API combines step 2 and 3 from the python examples into one request
- The API is also deployed in the test network and can be used for testing
- We highly recommend only using this API for making transaction on the production network
- The API supports making 3,000 request per minute / 50 request per second

Going live

- Just switch the SSN endpoint URI
- Change SSN network access identifyer
- Ready to deploy