
# Shioaji

## Usage Manual

Yvictor,

# Table of contents

# 1. Shioaji

shioaji-logo

永豐金證券 SinoPac Securities 永裕財富·豐盈人生



Shioaji Python Shioaji NumPy pandas PyTorch TensorFlow Python

:

- : C++ FPGA
- :
- : Python Python
- : Linux Python

## 1.1

### 1.1.1 Binaries

pip

```
pip install shioaji
```

shioaji

```
pip install -U shioaji
```

### 1.1.2 uv

uv

```
uv add shioaji
```

```
uv add shioaji --extra speed
```

### 1.1.3 Docker Image

Docker

```
docker run -it sinotrade/shioaji:latest
```

Jupyter Lab Jupyter Notebook

```
docker run -p 8888:8888 sinotrade/shioaji:jupyter
```

# 2.

Python

> **Token**

## 2.0.1

**version>=1.0** **version<1.0**

```
import shioaji as sj

api = sj.Shioaji()
accounts =  api.login("YOUR_API_KEY", "YOUR_SECRET_KEY")
api.activate_ca(
    ca_path="/c/your/ca/path/Sinopac.pfx",
    ca_passwd="YOUR_CA_PASSWORD",
    person_id="Person of this Ca",
)
```

```
import shioaji as sj

api = sj.Shioaji()
accounts = api.login("YOUR_PERSON_ID", "YOUR_PASSWORD")
api.activate_ca(
    ca_path="/c/your/ca/path/Sinopac.pfx",
    ca_passwd="YOUR_CA_PASSWORD",
    person_id="Person of this Ca",
)
```

> Windows `\` `/`

## 2.0.2

`subscribe`

```
api.quote.subscribe(api.Contracts.Stocks["2330"], quote_type="tick")
api.quote.subscribe(api.Contracts.Stocks["2330"], quote_type="bidask")
api.quote.subscribe(api.Contracts.Futures["TXFC0"], quote_type="tick")
```

> `shioaji.constent.QuoteType` `subscribe`

## 2.0.3

`place_order`

```
contract = api.Contracts.Stocks["2890"]
order = api.Order(
    price=12,
    quantity=5,
    action=sj.constant.Action.Buy,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
)
trade = api.place_order(contract, order)
```

## 2.1

Python Python

# 3.

## 3.1

- Windows MacOS Linux 64
- Python 3.8
- Shioaji API

## 3.2 Python

Python uv uv Python Shioaji API

uv python

### 3.2.1 uv

**Linux MacOS Windows**

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

```
powershell -c "irm https://astral.sh/uv/install.ps1 | iex"
```

uv

## 3.3

`sj-trading`

```
uv init sj-trading --package --app --vcs git
cd sj-trading
```

```
sj-trading
├── README.md
├── pyproject.toml
└── src
    └── sj_trading
        └── __init__.py
```

shioaji

```
uv add shioaji
```

`pyproject.toml`

```
[project]
name = "sj-trading"
version = "0.1.0"
description = "Shioaji Trading"
readme = "README.md"
requires-python = ">=3.12"
```

```
dependencies = [
    "shioaji>=1.2.5",
]

[project.scripts]
hello = "sj_trading:hello"

[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"
```

`hello` hello

**hello**

```
uv run hello
```

```
Hello from sj-trading!
```

`src/sj_trading/__init__.py`

```
import shioaji as sj

def hello():
    get_shioaji_client()


def get_shioaji_client() -> sj.Shioaji:
    api =  sj.Shioaji()
    print("Shioaji API created")
    return api
```

```
uv run hello
```

```
Shioaji API created
```

## 3.3.1 Jupyter

**ipykernel**

```
uv add --dev ipykernel
```

**Jupyter kernel**

```
uv run ipython kernel install --user --name=sj-trading
```

**Jupyter**

```
uv run --with jupyter jupyter lab
```

jupyter dev.ipynb `sj-trading` kernel

`hello`
jupyterlab

API Key

# 4. -

## 4.1

### 4.1.1

Shioaji :

1. open_acct
2. open_bank_1
3. **DAWHO+** open_bank_2
4.

## 4.1.2

1.0 Token

1. API
   newweb_1
2. API KEY
   newweb_2
3. API KEY
   newweb_3
4. API KEY IP
   newweb_4

- / : / API
- : API
- : API
- :

IP KEY

5. (API Key) (Secret Key)
   newweb_5

-
- Secret Key

1.

   newweb_7
2. API

   newweb_8

`sj-trading` `.env`

`.env`

```
API_KEY=<     API Key>
SECRET_KEY=<     Secret Key>
CA_CERT_PATH=<         >
CA_PASSWORD=<    >
```

```
sj-trading
├── README.md
├── .env
├── pyproject.toml
├── src
│   └── sj_trading
│       └── __init__.py
└── uv.lock
```

python-dotenv .env

```
uv add python-dotenv
```

`src/sj_trading/__init__.py`

```
import os
from dotenv import load_dotenv

load_dotenv()

def main():
    api = sj.Shioaji(simulation=True)
    api.login(
        api_key=os.environ["API_KEY"],
        secret_key=os.environ["SECRET_KEY"],
        fetch_contract=False
    )
    api.activate_ca(
        ca_path=os.environ["CA_CERT_PATH"],
        ca_passwd=os.environ["CA_PASSWORD"],
    )
    print("login and activate ca success")
```

`pyproject.toml` `[project.scripts]` `main`

```
[project.scripts]
main = "sj_trading:main"
```

`main`

```
uv run main
```

`login and activate ca success`

API

## 4.1.3

signature

**API**

:

- `login`
- `place_order`

**Attention**

:

- 08:00 ~ 20:00
- 18:00 ~ 20:00: IP
- 08:00 ~ 18:00:

:

- >= 1.2:
  : `uv add shioaji` or `pip install -U shioaji`

:

- API API
-
- / 1

```
import shioaji as sj

print(sj.__version__)
# 1.0.0
```

-

**version>=1.0** **version<1.0**

```
api = sj.Shioaji(simulation=True) #
api.login(
    api_key="  ",     #
    secret_key="  "   #
)

api = sj.Shioaji(simulation=True) #
api.login(
    person_id="     ", #
    passwd="  ",          #
)
```

- >= 1.0: `API Key` API Key Token
- < 1.0:

**version>=1.0** **version<1.0**

```
#     -
contract = api.Contracts.Stocks.TSE["2890"]

#       -
order = api.Order(
    price=18,                                   #
    quantity=1,                                 #
    action=sj.constant.Action.Buy,              #
    price_type=sj.constant.StockPriceType.LMT,  #
    order_type=sj.constant.OrderType.ROD,       #
    account=api.stock_account                   #
)

#
trade = api.place_order(contract, order)
trade

#     -
contract = api.Contracts.Stocks.TSE["2890"]

#       -
order = api.Order(
    price=18,                                   #
    quantity=1,                                 #
    action=sj.constant.Action.Buy,              #
    price_type=sj.constant.TFTStockPriceType.LMT,   #
    order_type=sj.constant.TFTOrderType.ROD,        #
    account=api.stock_account                   #
)

#
trade = api.place_order(contract, order)
trade
```

**Out**

```
Response Code: 0 | Event Code: 0 | Info: host '218.32.76.102:80', IP 218.32.76.102:80 (host 1 of 1) (host connection attempt 1 of 1) (total connection attempt 1
of 1) | Event: Session up

Trade(
    contract=Stock(...),
    order=Order(...),
    status=OrderStatus(
        id='531e27af',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 11, 18, 3, 867490),
        order_quantity=1,
        deals=[]
    )
)
```

- `Response Code: 0 | Event Code: 0 | Info: host '218.32.76.102:80' ...`

a. 

b. 

c. `signed`

- `Failed` `Failed` `place_order`
-
-

**verion>=1.0** **verion<1.0**

```
#     -       ,
contract = min(
    [
        x for x in api.Contracts.Futures.TXF
        if x.code[-2:] not in ["R1", "R2"]
    ],
    key=lambda x: x.delivery_date
)

#       -
order = api.Order(
    action=sj.constant.Action.Buy,                  #
    price=15000,                                    #
    quantity=1,                                     #
    price_type=sj.constant.FuturesPriceType.LMT,    #
    order_type=sj.constant.OrderType.ROD,           #
    octype=sj.constant.FuturesOCType.Auto,          #
    account=api.futopt_account                      #
)

#
trade = api.place_order(contract, order)
```

```
#     -       ,
contract = min(
    [
        x for x in api.Contracts.Futures.TXF
        if x.code[-2:] not in ["R1", "R2"]
    ],
    key=lambda x: x.delivery_date
)

#       -
order = api.Order(
    action=sj.constant.Action.Buy,                  #
    price=15000,                                    #
    quantity=1,                                     #
    price_type=sj.constant.FuturesPriceType.LMT,    #
    order_type=sj.constant.FuturesOrderType.ROD,    #
    octype=sj.constant.FuturesOCType.Auto,          #
    account=api.futopt_account                      #
)

#
trade = api.place_order(contract, order)
```

**Out**

```
Response Code: 0 | Event Code: 0 | Info: host '218.32.76.102:80', IP 218.32.76.102:80 (host 1 of 1) (host connection attempt 1 of 1) (total connection attempt 1 of 1) | Event: Session up
```

```
Trade(
    contract=Stock(...),
    order=Order(...),
    status=OrderStatus(
        id='531e27af',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 11, 18, 3, 867490),
        order_quantity=1,
        deals=[]
    )
)
```

- `Response Code: 0 | Event Code: 0 | Info: host '218.32.76.102:80' ...`

a. 

b. 

c. `signed`

- `Failed` `Failed` `place_order`
- 
- 

**API**

**Attention**

• API API

- 
- 
- API ( 5 )

**version>=1.0** **version<1.0**

```
import shioaji as sj

api = sj.Shioaji(simulation=False)  # Production Mode
accounts = api.login(
    api_key="YOUR_API_KEY",         # edit it
    secret_key="YOUR_SECRET_KEY"    # edit it
)
accounts
```

```
import shioaji as sj

api = sj.Shioaji(simulation=False)  # Production Mode
accounts = api.login(
    person_id="YOUR_PERSON_ID",     # edit it
    passwd="YOUR_PASSWORD",         # edit it
)
accounts
```

**Out**

```
Response Code: 0 | Event Code: 0 | Info: host '203.66.91.161:80', hostname '203.66.91.161:80' IP 203.66.91.161:80 (host 1 of 1) (host connection attempt 1 of 1)
(total connection attempt 1 of 1) | Event: Session up

[FutureAccount(person_id='QBCCAIGJBJ', broker_id='F002000', account_id='9100020', signed=True, username='PAPIUSER01'),
StockAccount(person_id='QBCCAIGJBJ', broker_id='9A95', account_id='0504350', username='PAPIUSER01')]
```

- `signed=True` : ! Ex: FutureAccount.
- `signed=False` `signed` : API API . Ex: StockAccount.

1. eleader
   eleader_download
2. eleader
   login eleader
3. (3303)
   eleader_account
4. " "
   CA_step
5. 
   CA_info

- 
- macOS docker shioaji

**In**

```
result = api.activate_ca(
    ca_path="/c/your/ca/path/Sinopac.pfx",
    ca_passwd="YOUR_CA_PASSWORD",
    person_id="Person ID of this Ca",
)

print(result)
# True
```

Windows \ /

**In**

```
api.get_ca_expiretime("Person ID")
```

## 4.1.4

`uv` `sj-trading`

sj-trading https://github.com/Sinotrade/sj-trading-demo

`git`

```
git clone https://github.com/Sinotrade/sj-trading-demo.git
cd sj-trading-demo
```

**SHIOAJI**

Shioaji

`src/sj_trading/__init__.py`

```
def show_version() -> str:
    print(f"Shioaji Version: {sj.__version__}")
    return sj.__version__
```

`pyproject.toml` `version`

```
[project.scripts]
version = "sj_trading:show_version"
```

`uv run version` Shioaji

```
Shioaji Version: 1.2.0
```

`src/sj_trading` `testing_flow.py`

```python
import shioaji as sj
from shioaji.constant import Action, StockPriceType, OrderType
import os

def testing_stock_ordering():
    #
    api = sj.Shioaji(simulation=True)
    accounts = api.login(
        api_key=os.environ["API_KEY"],
        secret_key=os.environ["SECRET_KEY"],
    )
    #
    print(f"Available accounts: {accounts}")
    api.activate_ca(
        ca_path=os.environ["CA_CERT_PATH"],
        ca_passwd=os.environ["CA_PASSWORD"],
    )

    #       Contract
    #    2890
    contract = api.Contracts.Stocks["2890"]
    print(f"Contract: {contract}")

    #         Order
    order = sj.order.StockOrder(
        action=Action.Buy, #
        price=contract.reference, #
        quantity=1, #
        price_type=StockPriceType.LMT, #
        order_type=OrderType.ROD, #
        account=api.stock_account, #
    )
    print(f"Order: {order}")

    #
    trade = api.place_order(contract=contract, order=order)
    print(f"Trade: {trade}")

    #
    api.update_status()
    print(f"Status: {trade.status}")
```

`pyproject.toml` `stock_testing`

```toml
[project.scripts]
stock_testing = "sj_trading.testing_flow:testing_stock_ordering"
```

`uv run stock_testing`

`src/sj_trading/testing_flow.py`

```
from shioaji.constant import (
    FuturesPriceType,
    FuturesOCType,
)

def testing_futures_ordering():
    #
    api = sj.Shioaji(simulation=True)
    accounts = api.login(
        api_key=os.environ["API_KEY"],
        secret_key=os.environ["SECRET_KEY"],
    )
    #
    print(f"Available accounts: {accounts}")
    api.activate_ca(
        ca_path=os.environ["CA_CERT_PATH"],
        ca_passwd=os.environ["CA_PASSWORD"],
    )

    #
    contract = api.Contracts.Futures["TXFR1"]
    print(f"Contract: {contract}")

    #           Order
    order = sj.order.FuturesOrder(
        action=Action.Buy,  #
        price=contract.reference,  #
        quantity=1,  #
        price_type=FuturesPriceType.LMT,  #
        order_type=OrderType.ROD,  #
        octype=FuturesOCType.Auto,  #
        account=api.futopt_account,  #
    )
    print(f"Order: {order}")

    #
    trade = api.place_order(contract=contract, order=order)
    print(f"Trade: {trade}")

    #
    api.update_status()
    print(f"Status: {trade.status}")
```

`pyproject.toml` `futures_testing`

```
[project.scripts]
futures_testing = "sj_trading.testing_flow:testing_futures_ordering"
```

`uv run futures_testing`

# 4.2

## 4.2.1

**Token login**

1.0 Token KEY 1.0

**In**

**version>=1.0** **version<1.0**

```
import shioaji as sj
api = sj.Shioaji()
api.login(
    api_key="YOUR_API_KEY",
    secret_key="YOUR_SECRET_KEY"
)

import shioaji as sj
api = sj.Shioaji()
api.login(
    person_id="YOUR_PERSON_ID",
    passwd="YOUR_PASSWORD",
)
```

**Out**

```
[FutureAccount(person_id='', broker_id='', account_id='', signed=True, username=''),
StockAccount(person_id='', broker_id='', account_id='', signed=True, username='')]
```

- If you cannot find `signed` in your accounts, please sign the document first.
- `signed` API

**Login Arguments**

**version>=1.0** **version<1.0**

```
api_key (str): API
secret_key (str):
fetch_contract (bool):                    (   : True)
contracts_timeout (int):       timeout (   : 0 ms)
contracts_cb (typing.Callable):       callback (   : None)
subscribe_trade (bool):       /     (   : True)
receive_window (int):            (   : 30,000   )

person_id (str):
passwd (str):
hashed (bool):        hashed (   : False)
fetch_contract (bool):                    (   : True)
contracts_timeout (int):       timeout (   : 0 ms)
contracts_cb (typing.Callable):       callback (   : None)
subscribe_trade (bool):       /     (   : True)
```

1.0 **Sign data is timeout** _____ _____ `receive_window`

### Callback

**In**

version>=1.0 version<1.0

```
import shioaji as sj
api = sj.Shioaji()
api.login(
    api_key="YOUR_API_KEY",
    secret_key="YOUR_SECRET_KEY",
    contracts_cb=lambda security_type: print(f"{repr(security_type)} fetch done.")
)

import shioaji as sj
api = sj.Shioaji()
api.login(
    person_id="YOUR_PERSON_ID",
    passwd="YOUR_PASSWORD",
    contracts_cb=lambda security_type: print(f"{repr(security_type)} fetch done.")
)
```

**Out**

```
[
    FutureAccount(person_id='', broker_id='', account_id='', signed=True, username=''),
    StockAccount(person_id='', broker_id='', account_id='', signed=True, username='')
]
<SecurityType.Index: 'IND'> fetch done.
<SecurityType.Future: 'FUT'> fetch done.
<SecurityType.Option: 'OPT'> fetch done.
<SecurityType.Stock: 'STK'> fetch done.
```

/

2 / `login` `subscribe_trade` `True` /

**In**

version>=1.0 version<1.0

```
import shioaji as sj
api = sj.Shioaji()
api.login(
    api_key="YOUR_API_KEY",
    secret_key="YOUR_SECRET_KEY",
    subscribe_trade=True
)

import shioaji as sj
api = sj.Shioaji()
api.login(
    person_id="YOUR_PERSON_ID",
    passwd="YOUR_PASSWORD",
    subscribe_trade=True
)
```

API `subscribe_trade` `unsubscribe_trade` /

**subscribe trade**

```
api.subscribe_trade(account)
```

**unsubscribe trade**

```
api.unsubscribe_trade(account)
```

## 4.2.2

In:

```
accounts = api.list_accounts()
accounts
```

Out

```
# print(accounts)
[
    FutureAccount(person_id='PERSON_ID_1', broker_id='BROKER_ID_1', account_id='ACCOUNT_ID_1', signed=True, username='USERNAME_1'),
    FutureAccount(person_id='PERSON_ID_2', broker_id='BROKER_ID_2', account_id='ACCOUNT_ID_2', username='USERNAME_2'),
    StockAccount(person_id='PERSON_ID_3', broker_id='BROKER_ID_3', account_id='ACCOUNT_ID_3', username='USERNAME_3'),
    StockAccount(person_id='PERSON_ID_4', broker_id='BROKER_ID_4', account_id='ACCOUNT_ID_4', signed=True, username='USERNAME_4')
]
```

- `signed` ACCOUNT_ID_2 ACCOUNT_ID_3

In

```
#
print(api.futopt_account)
```

Out

```
FutureAccount(person_id='PERSON_ID_1', broker_id='BROKER_ID_1', account_id='ACCOUNT_ID_1', signed=True, username='USERNAME_1')
```

In

```
#          ACCOUNT_ID_1    ACCOUNT_ID_2
api.set_default_account(accounts[1])
print(api.futopt_account)
```

Out

```
FutureAccount(person_id='PERSON_ID_2', broker_id='BROKER_ID_2', account_id='ACCOUNT_ID_2', username='USERNAME_2')
```

Order

```
order = api.Order(
    price=12,
    quantity=1,
    action=sj.constant.Action.Buy,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    order_lot=sj.constant.StockOrderLot.Common,
    account=api.stock_account
)
```

## 4.2.3

2021/08/06

```
api.logout()
# True
```

# 4.3

...

:

- 1: `Contracts.status` `contracts_timeout` 10000 10

**version>=1.0** **version<1.0**

```python
import shioaji as sj
api = sj.Shioaji()
api.login(
    api_key="YOUR_API_KEY",
    secret_key="YOUR_SECRET_KEY",
    contracts_timeout=10000,
)

import shioaji as sj
api = sj.Shioaji()
api.login(
    person_id="YOUR_PERSON_ID",
    passwd="YOUR_PASSWORD",
    contracts_timeout=10000,
)
```

- 2: `fetch_contract` `False` `fetch_contracts`

**version>=1.0** **version<1.0**

```python
import shioaji as sj
api = sj.Shioaji()
api.login(
    api_key="YOUR_API_KEY",
    secret_key="YOUR_SECRET_KEY",
    fetch_contract=False,
)
api.fetch_contracts(contract_download=True)

import shioaji as sj
api = sj.Shioaji()
api.login(
    person_id="YOUR_PERSON_ID",
    passwd="YOUR_PASSWORD",
    fetch_contract=False,
)
api.fetch_contracts(contract_download=True)
```

:

```python
api.Contracts
```

**Out**

```
Contracts(Indexs=(OTC, TSE), Stocks=(OES, OTC, TSE), Futures=(BRF, CAF, CBF, CCF, CDF, CEF, CFF, CGF, CHF, CJF, CK1, CKF, CLF, CM1, CMF, CNF, CQF, CRF, CSF, CU1,
CUF, CWF, CXF, CYF, CZ1, CZF, DCF, DD1, DDF, DEF, DFF, DGF, DHF, DIF, DJF, DKF, DLF, DNF, DOF, DPF, DQF, DSF, DUF, DVF, DWF, DXF, DYF, DZF, EEF, EGF, EHF, EMF,
EPF, ERF, EXF, EY1, EYF, FEF, FFF, FGF, FKF, FQF, FRF, FTF, FVF, FWF, FXF, FYF, FZF, G2F, GAF, GCF, GDF, GHF, GIF, GJF, GLF, GMF, GNF, GOF, GRF, GTF, GUF, GWF,
GXF, GZF, HAF, HBF, HCF, HHF, HIF, HLF, HOF, HS1, HSF, HY1, HYF, IA1, IAF, IHF, IIF, IJF, IMF, IOF, IPF, IQF, IRF, ITF, IXF, IYF, IZF, JBF, JFF, JNF, JPF, JSF,
JWF, JZF, KAF, KB1, KBF, KCF, KDF, KFF, KGF, KIF, KKF, KLF, KOF, KPF, KSF, KWF, LBF, LCF, LE1, LEF, LIF, LMF, LOF, LQF, LRF, LTF, LUF, LVF, LWF, LXF, LYF, MAF,
MBF, MCF, MJF, MKF, MPF, MQF, MVF, MX1, MXF, MYF, NAF, NBF, NCF, NDF, NEF, NGF, NHF, NIF, NJF, NLF, NMF, NNF, NOF, NQF, NSF, NUF, NVF, NWF, NXF, NYF, NZF, OAF,
OBF, OCF, ODF, OEF, OGF, OHF, OJF, OKF, OLF, OMF, OOF, OPF, OQF, ORF, OS1, OSF, OTF, OUF, OVF, OWF, OXF, OYF, OZF, PAF, PBF, PCF, PDF, PEF, PFF, PGF, PHF, PIF,
PJF, PKF, PLF, PMF, PNF, POF, PPF, PQF, RHF, RTF, SPF, T5F, TGF, TJF, TXF, UDF, UNF, XAF, XBF, XEF, XIF, XJF), Options=(CAO, CBO, CCO, CDO, CEO, CFO, CGO, CHO,
CJO, CKO, CLO, CMO, CNO, CQO, CRO, CSO, CXO, CZO, DCO, DEO, DFO, DGO, DHO, DJO, DKO, DLO, DNO, DOO, DPO, DQO, DSO, DUO, DVO, DWO, DXO, GIO, GXO, HCO, IJO, LOO,
NYA, NYO, NZO, OAO, OBO, OCO, OJO, OKO, OOO, OZO, RHO, RTO, TEO, TFO, TGO, TX1, TXO))
```

**In**

```
api.Contracts.Stocks["2890"]
# or api.Contracts.Stocks.TSE.TSE2890
```

**Out**

```
Stock(
    exchange=<Exchange.TSE: 'TSE'>,
    code='2890',
    symbol='TSE2890',
    name='    ',
    category='17',
    unit=1000,
    limit_up=19.1,
    limit_down=15.7,
    reference=17.4,
    update_date='2023/01/17',
    day_trade=<DayTrade.Yes: 'Yes'>
)
```

**Stock**

```
exchange (Exchange):     {OES, OTC, TSE ... }
code (str):
symbol (str):
name (str):
category (str):
unit (int):
limit_up (float):
limit_down (float):
reference (float):
update_date (str):
margin_trading_balance (int):
short_selling_balance (int):
day_trade (DayTrade):      {Yes, No, OnlyBuy}
```

**In**

```
api.Contracts.Futures["TXFA3"]
# or api.Contracts.Futures.TXF.TXF202301
```

**Out**

```
Future(
    code='TXFA3',
    symbol='TXF202301',
    name='    01',
    category='TXF',
    delivery_month='202301',
    delivery_date='2023/01/30',
    underlying_kind='I',
    unit=1,
    limit_up=16417.0,
    limit_down=13433.0,
    reference=14925.0,
    update_date='2023/01/18'
)
```

**Future**

```
code (str):
symbol (str):
name (str):
category (str):
delivery_month (str):
delivery_date (str):
underlying_kind (str):
unit (int):
limit_up (float):
limit_down (float):
reference (float):
update_date (str):
```

**In**

```
api.Contracts.Options["TXO18000R3"]
# or api.Contracts.Options.TXO.TXO20230618000P
```

**Out**

```
Option(
    code='TXO18000R3',
    symbol='TXO20230618000P',
    name='     06  18000P',
    category='TXO',
    delivery_month='202306',
    delivery_date='2023/06/21',
    strike_price=18000,
    option_right=<OptionRight.Put: 'P'>,
    underlying_kind='I',
    unit=1,
    limit_up=4720.0,
    limit_down=1740.0,
    reference=3230.0,
    update_date='2023/01/18'
)
```

**Option**

```
code (str):
symbol (str):
name (str):
category (str):
delivery_month (str):
delivery_date (str):
strike_price (int or float):
option_right (OptionRight):
underlying_kind (str):
limit_up (float):
limit_down (float):
reference (float):
update_date (str):
```

`Indexs`

**In**

```
api.Contracts.Indexs.TSE
```

**Out**

```
TSE(TSE001, TSE015, TSE016, TSE017, TSE018, TSE019, TSE020, TSE022, TSE023, TSE024, TSE025, TSE026, TSE028, TSE029, TSE030, TSE031, TSE032, TSE033, TSE035,
TSE036, TSE037, TSE038, TSE039, TSE040, TSE041, TSE042, TSE043, TSE004, TSE005)
```

**In**

```
api.Contracts.Indexs.TSE["001"]
# or api.Contracts.Indexs.TSE.TSE001
```

**Out**

```
Index(
    exchange=<Exchange.TSE: 'TSE'>,
    code='001',
    symbol='TSE001',
    name='    '
)
```

**Index**

```
exchange (Exchange):    {OES, OTC, TSE ... }
code (str):
symbol (str):
name (str):
```

# 4.4

## 4.4.1

**Subscribe**

```
>> api.quote.subscribe?

Signature:
    api.quote.subscribe(
        contract:shioaji.contracts.Contract,
        quote_type:shioaji.constant.QuoteType=<QuoteType.Tick: 'tick'>,
        intraday_odd:bool=False,
        version: shioaji.constant.QuoteVersion=<QuoteVersion.v0: 'v0'>,
    )
```

**Quote Parameters:**

```
quote_type:      {'tick', 'bidask'}
intraday_odd:      {True, False}
version:      {'v1', 'v0'}
```

**TICK**

**In**

```
api.quote.subscribe(
    api.Contracts.Stocks["2330"],
    quote_type = sj.constant.QuoteType.Tick,
    version = sj.constant.QuoteVersion.v1
)
```

**Out**

**QuoteVersion.v1** **QuoteVersion.v0**

```
Response Code: 200 | Event Code: 16 | Info: TIC/v1/STK/*/TSE/2330 | Event: Subscribe or Unsubscribe ok

Exchange.TSE
Tick(
    code = '2330',
    datetime = datetime.datetime(2021, 7, 2, 13, 16, 35, 92970),
    open = Decimal('590'),
    avg_price = Decimal('589.05'),
    close = Decimal('590'),
    high = Decimal('593'),
    low = Decimal('587'),
    amount = Decimal('590000'),
    total_amount = Decimal('8540101000'),
    volume = 1,
    total_volume = 14498,
    tick_type = 1,
    chg_type = 4,
    price_chg = Decimal('-3'),
    pct_chg = Decimal('-0.505902'),
    bid_side_total_vol= 6638,
    ask_side_total_vol = 7860,
    bid_side_total_cnt = 2694,
    ask_side_total_cnt = 2705,
    closing_oddlot_shares = 0,
    fixed_trade_vol = 0,
    suspend = 0,
    simtrade = 0,
    intraday_odd = 0
)

Response Code: 200 | Event Code: 16 | Info: MKT/*/TSE/2330 | Event: Subscribe or Unsubscribe ok

MKT/idcdmzpcr01/TSE/2330
{
    'AmountSum': [1688787000.0],
    'Close': [593.0],
    'Date': '2021/07/01',
    'TickType': [2],
    'Time': '09:10:20.628620',
    'VolSum': [2837],
    'Volume': [1]
}
```

**In**

```
api.quote.subscribe(
    api.Contracts.Stocks["2330"],
    quote_type = sj.constant.QuoteType.Tick,
    version = sj.constant.QuoteVersion.v1,
    intraday_odd = True
)
```

### Out

**QuoteVersion.v1** **QuoteVersion.v0**

```
Response Code: 200 | Event Code: 16 | Info: TIC/v1/ODD/*/TSE/2330 | Event: Subscribe or Unsubscribe ok

Exchange.TSE
Tick(
    code = '2330',
    datetime = datetime.datetime(2021, 7, 2, 13, 16, 55, 544646),
    open = Decimal('591'),
    avg_price = Decimal('590.24415'),
    close = Decimal('590'),
    high = Decimal('591'),
    low = Decimal('589'),
    amount = Decimal('276120'),
    total_amount = Decimal('204995925'),
    volume = 468,
    total_volume = 347307,
    tick_type = 1,
    chg_type = 4,
    price_chg = Decimal('-3'),
    pct_chg = Decimal('-0.505902'),
    bid_side_total_vol= 68209,
    ask_side_total_vol = 279566,
    bid_side_total_cnt = 28,
    ask_side_total_cnt = 56,
    closing_oddlot_shares = 0,
    fixed_trade_vol = 0,
    suspend = 0,
    simtrade = 1,
    intraday_odd = 1
)

Response Code: 200 | Event Code: 16 | Info: TIC/v2/*/TSE/2330/ODDLOT | Event: Subscribe or Unsubscribe ok

TIC/v2/replay/TSE/2330/ODDLOT
{
    'Date': '2021/07/01',
    'Time': '09:23:36.880878',
    'Close': '593',
    'TickType': 1,
    'Shares': 1860,
    'SharesSum': 33152,
    'Simtrade': 1
}
```

### Tick

**QuoteVersion.v1** **QuoteVersion.v0**

```
code (str):
datetime (datetime):
open (decimal):
avg_price (decimal):
close (decimal):
high (decimal):    (   )
low (decimal):    (   )
amount (decimal):     (NTD)
total_amount (decimal):      (NTD)
volume (int):     (  : ,     :  )
total_volume (int):      (  : ,     :  )
tick_type (int):     {1:   , 2:   , 0:     }
chg_type (int):     {1:   , 2:  , 3:   , 4:  , 5:   }
price_chg (decimal):
pct_chg (decimal):
bid_side_total_vol (int):        (  : ,     :  )
ask_side_total_vol (int):        (  : ,     :  )
bid_side_total_cnt (int):
ask_side_total_cnt (int):
closing_oddlot_shares (int):         ( )
fixed_trade_vol (int):       (  : ,     :  )
suspend (bool):
simtrade (bool):
intraday_odd (int):      {0:   , 1:    }

AmountSum (:List:float):
Close (:List:float):
Date (str):    (yyyy/MM/dd)
TickType (:List:int):     {1:   , 2:   , 0:     }
Time (str):    (HH:mm:ss.ffffff)
VolSum (:List:int):      ( )
Volume (:List:int):     ( )
```

### BIDASK

In

```
api.quote.subscribe(
    api.Contracts.Stocks["2330"],
    quote_type = sj.constant.QuoteType.BidAsk,
    version = sj.constant.QuoteVersion.v1
)
```

Out

QuoteVersion.v1 QuoteVersion.v0

```
Response Code: 200 | Event Code: 16 | Info: QUO/v1/STK/*/TSE/2330 | Event: Subscribe or Unsubscribe ok

Exchange.TSE
BidAsk(
    code = '2330',
    datetime = datetime.datetime(2021, 7, 1, 9, 9, 54, 36828),
    bid_price = [Decimal('593'), Decimal('592'), Decimal('591'), Decimal('590'), Decimal('589')],
    bid_volume = [248, 180, 258, 267, 163],
    diff_bid_vol = [3, 0, 0, 0, 0],
    ask_price = [Decimal('594'), Decimal('595'), Decimal('596'), Decimal('597'), Decimal('598')],
    ask_volume = [1457, 531, 506, 90, 259],
    diff_ask_vol = [0, 0, 0, 0, 0],
    suspend = 0,
    simtrade = 0,
    intraday_odd = 0
)

Response Code: 200 | Event Code: 16 | Info: QUT/*/TSE/2330 | Event: Subscribe or Unsubscribe ok

QUT/idcdmzpcr01/TSE/2330
{
    'AskPrice': [594.0, 595.0, 596.0, 597.0, 598.0],
    'AskVolume': [1465, 532, 507, 92, 258],
    'BidPrice': [593.0, 592.0, 591.0, 590.0, 589.0],
    'BidVolume': [254, 178, 255, 268, 163],
    'Date': '2021/07/01',
    'Time': '09:09:48.447219'
}
```

In

```
api.quote.subscribe(
    api.Contracts.Stocks["2330"],
    quote_type = sj.constant.QuoteType.BidAsk,
    version = sj.constant.QuoteVersion.v1
    intraday_odd=True
)
```

**Out**

**QuoteVersion.v1** **QuoteVersion.v0**

```
Response Code: 200 | Event Code: 16 | Info: QUO/v1/ODD/*/TSE/2330 | Event: Subscribe or Unsubscribe ok

Exchange.TSE
BidAsk(
    code = '2330',
    datetime = datetime.datetime(2021, 7, 2, 13, 17, 45, 743299),
    bid_price = [Decimal('589'), Decimal('588'), Decimal('587'), Decimal('586'), Decimal('585')],
    bid_volume = [59391, 224490, 74082, 68570, 125246],
    diff_bid_vol = [49874, 101808, 23863, 38712, 77704],
    ask_price = [Decimal('590'), Decimal('591'), Decimal('592'), Decimal('593'), Decimal('594')],
    ask_volume = [26355, 9680, 18087, 11773, 3568],
    diff_ask_vol = [13251, -14347, 39249, -20397, -10591],
    suspend = 0,
    simtrade = 1,
    intraday_odd = 1
)

Response Code: 200 | Event Code: 16 | Info: QUO/v2/*/TSE/2330/ODDLOT | Event: Subscribe or Unsubscribe ok

QUO/v2/replay/TSE/2330/ODDLOT
{
    'Date': '2021/07/01',
    'Time': '09:43:47.143789',
    'BidPrice': ['592', '591', '590', '589', '588'],
    'AskPrice': ['593', '594', '595', '596', '597'],
    'BidShares': [16979, 12009, 45045, 5501, 12956],
    'AskShares': [17276, 14823, 26518, 23388, 10527],
    'Simtrade': 1
}
```

**BidAsk**

**QuoteVersion.v1** **QuoteVersion.v0**

```
code (str):
datetime (datetime):
bid_price (:List:decimal):
bid_volume (:List:int):     ( )
diff_bid_vol (:List:int):       ( )
ask_price (:List:decimal):
ask_volume (:List:int):
diff_ask_vol (:List:int):       ( )
suspend (bool):
simtrade (bool):

AskPrice (:List:float):
AskVolume (:List:int):
BidPrice (:List:float):
BidVolume (:List:int):
Date (datetime.date):    (yyyy/MM/dd)
Time (time):    (HH:mm:ss.ffffff)
```

**QUOTE**

**In**

```
api.quote.subscribe(
    api.Contracts.Stocks["2330"],
    quote_type = sj.constant.QuoteType.Quote,
    version = sj.constant.QuoteVersion.v1
)
```

**Out**

```
Response Code: 200 | Event Code: 16 | Info: QUO/v2/STK/*/TSE/2330 | Event: Subscribe or Unsubscribe ok

Exchange.TSE,
Quote(
    code='2330',
    datetime=datetime.datetime(2022, 7, 1, 10, 43, 15, 430329),
    open=Decimal('471.5'),
    avg_price=Decimal('467.91'),
    close=Decimal('461'),
    high=Decimal('474'),
    low=Decimal('461'),
    amount=Decimal('461000'),
    total_amount=Decimal('11834476000'),
    volume=1,
    total_volume=25292,
    tick_type=2,
    chg_type=4,
    price_chg=Decimal('-15'),
    pct_chg=Decimal('-3.15'),
    bid_side_total_vol=9350,
    ask_side_total_vol=15942,
    bid_side_total_cnt=2730,
    ask_side_total_cnt=2847,
    closing_oddlot_shares=0,
    closing_oddlot_close=Decimal('0.0'),
    closing_oddlot_amount=Decimal('0'),
    closing_oddlot_bid_price=Decimal('0.0'),
    closing_oddlot_ask_price=Decimal('0.0'),
    fixed_trade_vol=0,
    fixed_trade_amount=Decimal('0'),
    bid_price=[Decimal('461'), Decimal('460.5'), Decimal('460'), Decimal('459.5'), Decimal('459')],
    bid_volume=[220, 140, 994, 63, 132],
    diff_bid_vol=[-1, 0, 0, 0, 0],
    ask_price=[Decimal('461.5'), Decimal('462'), Decimal('462.5'), Decimal('463'), Decimal('463.5')],
    ask_volume=[115, 101, 103, 147, 91],
    diff_ask_vol=[0, 0, 0, 0, 0],
    avail_borrowing=9579699,
    suspend=0,
    simtrade=0
)
```

**Quote**

```
code (str):
datetime (datetime):
open (decimal):
avg_price (decimal):
close (decimal):
high (decimal):    (   )
low (decimal):    (   )
amount (decimal):     (NTD)
total_amount (decimal):      (NTD)
volume (int):
total_volume (int):
tick_type (int):
chg_type (int):
price_chg (decimal):
pct_chg (decimal):
bid_side_total_vol (int):        ( )
ask_side_total_vol (int):        ( )
bid_side_total_cnt (int):
ask_side_total_cnt (int):
closing_oddlot_shares (int):
closing_oddlot_close (decimal):
closing_oddlot_amount (decimal):
closing_oddlot_bid_price (decimal):
closing_oddlot_ask_price (decimal):
fixed_trade_vol (int):       ( )
fixed_trade_amount (decimal):
bid_price (:List:decimal):
bid_volume (:List:int)
diff_bid_vol (:List:int)
ask_price (:List:decimal):
ask_volume (:List:int)
diff_ask_vol (:List:int)
avail_borrowing (int):
suspend (bool):
simtrade (bool):
```

**CALLBACK**

`print`

Tick

**decorator**

QuoteVersion.v1 QuoteVersion.v0

```python
from shioaji import TickSTKv1, Exchange

@api.on_tick_stk_v1()
def quote_callback(exchange: Exchange, tick:TickSTKv1):
    print(f"Exchange: {exchange}, Tick: {tick}")

@api.quote.on_quote
def quote_callback(topic: str, quote: dict):
    print(f"Topic: {topic}, Quote: {quote}")
```

QuoteVersion.v1 QuoteVersion.v0

```python
from shioaji import TickSTKv1, Exchange

def quote_callback(exchange: Exchange, tick:TickSTKv1):
    print(f"Exchange: {exchange}, Tick: {tick}")

api.quote.set_on_tick_stk_v1_callback(quote_callback)

def quote_callback(topic: str, quote: dict):
    print(f"Topic: {topic}, Quote: {quote}")

api.quote.set_quote_callback(quote_callback)
```

**Out**

QuoteVersion.v1 QuoteVersion.v0

```
Exchange: Exchange.TSE, Tick: Tick(code='2330', datetime=datetime.datetime(2021, 7, 2, 13, 16, 35, 92970), open=Decimal('590'), avg_price=Decimal('589.05'),
close=Decimal('590'), high=Decimal('593'), low=Decimal('587'), amount=Decimal('590000'), total_amount=Decimal('8540101000'), volume=1, total_volume=14498,
tick_type=1, chg_type=4, price_chg=Decimal('-3'), pct_chg=Decimal('-0.505902'), trade_bid_volume=6638, ask_side_total_vol=7860, bid_side_total_cnt=2694,
ask_side_total_cnt=2705, closing_oddlot_shares=0, fixed_trade_vol=0, suspend=0, simtrade=0, intraday_odd=0)

Topic: MKT/*/TSE/2330, Quote: {'AmountSum': [4739351000.0], 'Close': [596.0], 'Date': '2021/03/30', 'TickType': [2], 'Time': '10:01:33.349431', 'VolSum': [7932],
'Volume': [1]}
```

**BidAsk**

**decorator**

QuoteVersion.v1 QuoteVersion.v0

```python
from shioaji import BidAskSTKv1, Exchange

@api.on_bidask_stk_v1()
def quote_callback(exchange: Exchange, bidask:BidAskSTKv1):
    print(f"Exchange: {exchange}, BidAsk: {bidask}")

@api.quote.on_quote
def quote_callback(topic: str, quote: dict):
    print(f"Topic: {topic}, Quote: {quote}")
```

**QuoteVersion.v1** **QuoteVersion.v0**

```
from shioaji import BidAskSTKv1, Exchange

def quote_callback(exchange: Exchange, bidask:BidAskSTKv1):
    print(f"Exchange: {exchange}, BidAsk: {bidask}")

api.quote.set_on_bidask_stk_v1_callback(quote_callback)

def quote_callback(topic: str, quote: dict):
    print(f"Topic: {topic}, Quote: {quote}")

api.quote.set_quote_callback(quote_callback)
```

**Out**

**QuoteVersion.v1** **QuoteVersion.v0**

```
Exchange: Exchange.TSE, BidAsk: BidAsk(code='2330', datetime=datetime.datetime(2021, 7, 2, 13, 17, 29, 726428), bid_price=[Decimal('589'), Decimal('588'),
Decimal('587'), Decimal('586'), Decimal('585')], bid_volume=[223, 761, 1003, 809, 1274], diff_bid_vol=[0, 0, 0, 0, 0], ask_price=[Decimal('590'), Decimal('591'),
Decimal('592'), Decimal('593'), Decimal('594')], ask_volume=[304, 232, 183, 242, 131], diff_ask_vol=[1, 0, 0, 0, 0], suspend=0, simtrade=0, intraday_odd=0)

Topic: QUT/idcdmzpcr01/TSE/2330, Quote: {'AskPrice': [590.0, 591.0, 592.0, 593.0, 594.0], 'AskVolume': [303, 232, 183, 242, 131], 'BidPrice': [589.0, 588.0,
587.0, 586.0, 585.0], 'BidVolume': [224, 762, 1003, 809, 1274], 'Date': '2021/07/02', 'Time': '13:17:26.391840'}
```

**Quote**

**decorator**

```
from shioaji import QuoteSTKv1, Exchange

@api.on_quote_stk_v1()
def quote_callback(exchange: Exchange, quote:QuoteSTKv1):
    print(f"Exchange: {exchange}, Quote: {quote}")
```

```
from shioaji import QuoteSTKv1, Exchange

def quote_callback(exchange: Exchange, quote:QuoteSTKv1):
    print(f"Exchange: {exchange}, Quote: {quote}")

api.quote.set_on_quote_stk_v1_callback(quote_callback)
```

**Out**

```
Exchange: TSE, Quote: Quote(code='2330', datetime=datetime.datetime(2022, 7, 1, 10, 43, 15, 430329), open=Decimal('471.5'), avg_price=Decimal('467.91'),
close=Decimal('461'), high=Decimal('474'), low=Decimal('461'), amount=Decimal('461000'), total_amount=Decimal('11834476000'), volume=1, total_volume=25292,
tick_type=2, chg_type=4, price_chg=Decimal('-15'), pct_chg=Decimal('-3.15'), bid_side_total_vol=9350, ask_side_total_vol=15942, bid_side_total_cnt=2730,
ask_side_total_cnt=2847, closing_oddlot_shares=0, closing_oddlot_close=Decimal('0.0'), closing_oddlot_amount=Decimal('0'),
closing_oddlot_bid_price=Decimal('0.0'), closing_oddlot_ask_price=Decimal('0.0'), fixed_trade_vol=0, fixed_trade_amount=Decimal('0'), bid_price=[Decimal('461'),
Decimal('460.5'), Decimal('460'), Decimal('459.5'), Decimal('459')], bid_volume=[220, 140, 994, 63, 132], diff_bid_vol=[-1, 0, 0, 0, 0],
ask_price=[Decimal('461.5'), Decimal('462'), Decimal('462.5'), Decimal('463'), Decimal('463.5')], ask_volume=[115, 101, 103, 147, 91], diff_ask_vol=[0, 0, 0, 0,
0], avail_borrowing=9579699, suspend=0, simtrade=0)
```

- callback
- callback

### Subscribe

```
api.quote.subscribe?

    Signature:
        api.quote.subscribe(
            contract:shioaji.contracts.Contract,
            quote_type:shioaji.constant.QuoteType=<QuoteType.Tick: 'tick'>,
            intraday_odd:bool=False,
            version: shioaji.constant.QuoteVersion=<QuoteVersion.v0: 'v0'>,
        )
    Docstring: <no docstring>
    Type:      method
```

### Quote Parameters:

```
quote_type:      {'tick', 'bidask'}
intraday_odd:      {True, False}
version:      {'v1', 'v0'}
```

**TICK**

```
api.quote.subscribe(
    api.Contracts.Futures.TXF['TXF202107'],
    quote_type = sj.constant.QuoteType.Tick,
    version = sj.constant.QuoteVersion.v1,
)
```

**Out**

**QuoteVersion.v1** **QuoteVersion.v0**

```
Response Code: 200 | Event Code: 16 | Info: TIC/v1/FOP/*/TFE/TXFG1 | Event: Subscribe or Unsubscribe ok

Exchange.TAIFEX
Tick(
    code = 'TXFG1',
    datetime = datetime.datetime(2021, 7, 1, 10, 42, 29, 757000),
    open = Decimal('17678'),
    underlying_price = Decimal('17849.57'),
    bid_side_total_vol= 32210,
    ask_side_total_vol= 33218,
    avg_price = Decimal('17704.663999'),
    close = Decimal('17753'),
    high = Decimal('17774'),
    low = Decimal('17655'),
    amount = Decimal('17753'),
    total_amount = Decimal('913790823'),
    volume = 1,
    total_volume = 51613,
    tick_type = 0,
    chg_type = 2,
    price_chg = Decimal('41'),
    pct_chg = Decimal('0.231481'),
    simtrade = 0
)

Response Code: 200 | Event Code: 16 | Info: L/*/TXFG1 | Event: Subscribe or Unsubscribe ok

L/TFE/TXFG1
{
    'Amount': [17754.0],
    'AmountSum': [913027415.0],
    'AvgPrice': [17704.623134],
    'Close': [17754.0],
    'Code': 'TXFG1',
    'Date': '2021/07/01',
    'DiffPrice': [42.0],
    'DiffRate': [0.237127],
    'DiffType': [2],
    'High': [17774.0],
    'Low': [17655.0],
    'Open': 17678.0,
    'TargetKindPrice': 17849.57,
    'TickType': [2],
    'Time': '10:42:25.552000',
    'TradeAskVolSum': 33198,
    'TradeBidVolSum': 32180,
    'VolSum': [51570],
    'Volume': [1]
}
```

### Tick

**QuoteVersion.v1** **QuoteVersion.v0**

```
code (str):
datetime (datetime.datetime):
open (Decimal):
underlying_price (Decimal):
bid_side_total_vol(int):        (lot)
ask_side_total_vol(int):        (lot)
avg_price (Decimal):
close (Decimal):
high (Decimal):    (   )
low (Decimal):    (   )
amount (Decimal):     (NTD)
total_amount (Decimal):      (NTD)
volume (int):     (lot)
total_volume (int):      (lot)
tick_type (int):     {1:   , 2:   , 0:     }
chg_type (int):     {1:   , 2:  , 3:   , 4:  , 5:   }
price_chg (Decimal):
pct_chg (Decimal):     (%)
simtrade (int):

Amount (list of float):     (   )
AmountSum (list of float):      (    )
AvgPrice (list of float):
Close (list of float):
Code (str):
Date (str):    (yyyy/MM/dd)
DiffPrice (list of float):
DiffRate (list of float):     (%)
DiffType (list of int):     {1:   , 2:  , 3:   , 4:  , 5:   }
High (list of float):    (   )
Low (list of float):    (   )
Open (float):
TargetKindPrice float:
TickType (:list:int):     {1:   , 2:   , 0:     }
Time (str):    (HH:mm:ss.ffffff)
TradeAskVolSum (int):        (lot)
TradeBidVolSum (int):        (lot)
VolSum (list of int):      (lot)
Volume (list of int):     (lot)
```

**BIDASK**

```
api.quote.subscribe(
    api.Contracts.Futures.TXF['TXF202107'],
    quote_type = sj.constant.QuoteType.BidAsk,
    version = sj.constant.QuoteVersion.v1
)
```

**Out**

**QuoteVersion.v1** **QuoteVersion.v0**

```
Response Code: 200 | Event Code: 16 | Info: QUO/v1/FOP/*/TFE/TXFG1 | Event: Subscribe or Unsubscribe ok

Exchange.TAIFEX
BidAsk(
    code = 'TXFG1',
    datetime = datetime.datetime(2021, 7, 1, 10, 51, 31, 999000),
    bid_total_vol = 66,
    ask_total_vol = 101,
    bid_price = [Decimal('17746'), Decimal('17745'), Decimal('17744'), Decimal('17743'), Decimal('17742')],
    bid_volume = [1, 14, 19, 17, 15],
    diff_bid_vol = [0, 1, 0, 0, 0],
    ask_price = [Decimal('17747'), Decimal('17748'), Decimal('17749'), Decimal('17750'), Decimal('17751')],
    ask_volume = [6, 22, 25, 32, 16],
    diff_ask_vol = [0, 0, 0, 0, 0],
    first_derived_bid_price = Decimal('17743'),
    first_derived_ask_price = Decimal('17751'),
    first_derived_bid_vol = 1,
    first_derived_ask_vol = 1,
    underlying_price = Decimal('17827.94'),
    simtrade = 0
)

Response Code: 200 | Event Code: 16 | Info: Q/*/TXFG1 | Event: Subscribe or Unsubscribe ok

Q/TFE/TXFG1
{
    'AskPrice': [17747.0, 17748.0, 17749.0, 17750.0, 17751.0],
    'AskVolSum': 99,
    'AskVolume': [6, 22, 25, 31, 15],
    'BidPrice': [17746.0, 17745.0, 17744.0, 17743.0, 17742.0],
    'BidVolSum': 81,
    'BidVolume': [1, 12, 23, 25, 20],
    'Code': 'TXFG1',
    'Date': '2021/07/01',
    'DiffAskVol': [0, 0, 0, 0, 0],
    'DiffAskVolSum': 0,
    'DiffBidVol': [0, 0, 2, 0, 0],
    'DiffBidVolSum': 0,
    'FirstDerivedAskPrice': 17751.0,
    'FirstDerivedAskVolume': 1,
    'FirstDerivedBidPrice': 17743.0,
    'FirstDerivedBidVolume': 1,
    'TargetKindPrice': 17828.46,
    'Time': '10:51:29.999000'
}
```

**BidAsk**

QuoteVersion.v1 QuoteVersion.v0

```
code (str):
datetime (datetime.datetime):
bid_total_vol (int):       (lot)
ask_total_vol (int):       (lot)
bid_price (:List:decimal):
bid_volume (:List:int):     (lot)
diff_bid_vol (:List:int):        (lot)
ask_price (:List:decimal):
ask_volume (:List:int):     (lot)
diff_ask_vol (:List:int):        (lot)
first_derived_bid_price (decimal):
first_derived_ask_price (decimal):
first_derived_bid_vol (int):
first_derived_ask_vol (int):
underlying_price (decimal):
simtrade (int):

AskPrice (:List:float):
AskVolSum (int):      (lot)
AskVolume (:List:int):
BidPrice (:List:float):
BidVolSum (int):      (lot)
BidVolume (:List:int):
Code (str):
Date (str):    (yyyy/MM/dd)
DiffAskVol (:List:int):      (lot)
DiffAskVolSum (int):
DiffBidVol (:List:int):      (lot)
DiffBidVolSum (int):
FirstDerivedAskPrice (float):
FirstDerivedAskVolume (int):
FirstDerivedBidPrice (float):
FirstDerivedBidVolume (int):
TargetKindPrice (float):
Time (str):    (HH:mm:ss.ffffff)
```

**CALLBACK**

`print`

**Tick**

**decorator**

QuoteVersion.v1 QuoteVersion.v0

```python
from shioaji import TickFOPv1, Exchange

@api.on_tick_fop_v1()
def quote_callback(exchange:Exchange, tick:TickFOPv1):
    print(f"Exchange: {exchange}, Tick: {tick}")

@api.quote.on_quote
def quote_callback(topic: str, quote: dict):
    print(f"Topic: {topic}, Quote: {quote}")
```

QuoteVersion.v1 QuoteVersion.v0

```python
from shioaji import TickFOPv1, Exchange

def quote_callback(exchange:Exchange, tick:TickFOPv1):
    print(f"Exchange: {exchange}, Tick: {tick}")

api.quote.set_on_tick_fop_v1_callback(quote_callback)

def quote_callback(topic: str, tick: dict):
    print(f"Topic: {topic}, Tick: {tick}")

api.quote.set_quote_callback(quote_callback)
```

**Out**

QuoteVersion.v1 QuoteVersion.v0

```
Exchange: Exchange.TAIFEX, Tick: Tick(code='TXFG1', datetime=datetime.datetime(2021, 7, 2, 13, 17, 22, 784000), open=Decimal('17651'),
underlying_price=Decimal('17727.12'), trade_bid_total_vol=61550, trade_ask_volume=60914, avg_price=Decimal('17657.959752'), close=Decimal('17653'),
high=Decimal('17724'), low=Decimal('17588'), amount=Decimal('35306'), total_amount=Decimal('1683421593'), volume=2, total_volume=95335, tick_type=1, chg_type=2,
price_chg=Decimal('7'), pct_chg=Decimal('0.039669'), simtrade=0)

Topic: L/TFE/TXFG1, Quote: {'Amount': [17654.0], 'AmountSum': [1682856730.0], 'AvgPrice': [17657.961764], 'Close': [17654.0], 'Code': 'TXFG1', 'Date':
'2021/07/02', 'DiffPrice': [8.0], 'DiffRate': [0.045336], 'DiffType': [2], 'High': [17724.0], 'Low': [17588.0], 'Open': 17651.0, 'TargetKindPrice': 17725.14,
'TickType': [1], 'Time': '13:17:16.533000', 'TradeAskVolSum': 60890, 'TradeBidVolSum': 61520, 'VolSum': [95303], 'Volume': [1]}
```

**BidAsk**

**decorator**

QuoteVersion.v1 QuoteVersion.v0

```
from shioaji import BidAskFOPv1, Exchange

@api.on_bidask_fop_v1()
def quote_callback(exchange:Exchange, bidask:BidAskFOPv1):
    print(f"Exchange: {exchange}, BidAsk: {bidask}")

@api.quote.on_quote
def quote_callback(topic: str, quote: dict):
    print(f"Topic: {topic}, Quote: {quote}")
```

QuoteVersion.v1 QuoteVersion.v0

```
from shioaji import BidAskFOPv1, Exchange

def quote_callback(exchange:Exchange, bidask:BidAskFOPv1):
    print(f"Exchange: {exchange}, BidAsk: {bidask}")

api.quote.set_on_bidask_fop_v1_callback(quote_callback)

def quote_callback(topic: str, quote: dict):
    print(f"Topic: {topic}, Quote: {quote}")

api.quote.set_quote_callback(quote_callback)
```

**Out**

QuoteVersion.v1 QuoteVersion.v0

```
Exchange: Exchange.TAIFEX, BidAsk: BidAsk(code='TXFG1', datetime=datetime.datetime(2021, 7, 2, 13, 18, 0, 684000), bid_total_vol=69, ask_total_vol=94,
bid_price=[Decimal('17651'), Decimal('17650'), Decimal('17649'), Decimal('17648'), Decimal('17647')], bid_volume=[10, 12, 18, 18, 11], diff_bid_vol=[0, 0, 0, 0,
0], ask_price=[Decimal('17653'), Decimal('17654'), Decimal('17655'), Decimal('17656'), Decimal('17657')], ask_volume=[6, 17, 29, 22, 20], diff_ask_vol=[0, 0, 0,
0, 0], first_derived_bid_price=Decimal('17647'), first_derived_ask_price=Decimal('17657'), first_derived_bid_vol=2, first_derived_ask_vol=3,
underlying_price=Decimal('17725.5'), simtrade=0)

Topic: Q/TFE/TXFG1, Quote: {'AskPrice': [17653.0, 17654.0, 17655.0, 17656.0, 17657.0], 'AskVolSum': 85, 'AskVolume': [3, 16, 24, 22, 20], 'BidPrice': [17651.0,
17650.0, 17649.0, 17648.0, 17647.0], 'BidVolSum': 67, 'BidVolume': [10, 10, 18, 18, 11], 'Code': 'TXFG1', 'Date': '2021/07/02', 'DiffAskVol': [-4, -2, 0, 0, 0],
'DiffAskVolSum': 0, 'DiffBidVol': [1, 0, 2, 0, 0], 'DiffBidVolSum': 0, 'FirstDerivedAskPrice': 17657.0, 'FirstDerivedAskVolume': 3, 'FirstDerivedBidPrice':
17647.0, 'FirstDerivedBidVolume': 2, 'TargetKindPrice': 17716.19, 'Time': '13:17:57.809000'}
```

- callback

## 4.4.2

### Ticks

Ticks

**Ticks**

```
api.ticks?

Signature:
    api.ticks(
        contract: shioaji.contracts.BaseContract,
        date: str = '2022-12-26',
        query_type: shioaji.constant.TicksQueryType = <TicksQueryType.AllDay: 'AllDay'>,
        time_start: Union[str, datetime.time] = None,
        time_end: Union[str, datetime.time] = None,
        last_cnt: int = 0,
        timeout: int = 30000,
        cb: Callable[[shioaji.data.Ticks], NoneType] = None,
    ) -> shioaji.data.Ticks
Docstring:
    get contract tick volumn
```

TICKS

**In**

```
ticks = api.ticks(
    contract=api.Contracts.Stocks["2330"],
    date="2023-01-16"
)
ticks
```

**Out**

```
Ticks(
    ts=[1673859600113699000, 1673859600228800000, 1673859600244294000, 1673859600308595000],
    close=[506.0, 505.0, 506.0, 506.0],
    volume=[3340, 1, 17, 2],
    bid_price=[505.0, 505.0, 506.0, 506.0],
    bid_volume=[122, 320, 60, 58],
    ask_price=[506.0, 506.0, 507.0, 507.0],
    ask_volume=[13, 22, 702, 702]
    tick_type=[1, 2, 1, 2]
)
```

**Ticks**

```
ts (int): timestamp
close (float):
volume (int):
bid_price (float):
bid_volume (int):
ask_price (float):
ask_volume (int):
tick_type (int):    {1:   , 2:   , 0:    }
```

DataFrame

**In**

```
import pandas as pd
df = pd.DataFrame({**ticks})
df.ts = pd.to_datetime(df.ts)
df.head()
```

Out

| ts | ask_price | close | bid_volume | volume | ask_volume | tick_type | bid_price |
|---|---|---|---|---|---|---|---|
| 2023-01-16 09:00:00.113699 | 506 | 506 | 122 | 3340 | 13 | 1 | 505 |
| 2023-01-16 09:00:00.228800 | 506 | 505 | 320 | 1 | 22 | 2 | 505 |
| 2023-01-16 09:00:00.244294 | 507 | 506 | 60 | 17 | 702 | 1 | 506 |
| 2023-01-16 09:00:00.308595 | 507 | 506 | 58 | 2 | 702 | 2 | 506 |

TICKS

In

```
ticks = api.ticks(
    contract=api.Contracts.Stocks["2330"],
    date="2023-01-16",
    query_type=sj.constant.TicksQueryType.RangeTime,
    time_start="09:00:00",
    time_end="09:20:01"
)
ticks
```

Out

```
Ticks(
    ts=[1673859600113699000, 1673859600228800000, 1673859600244294000, 1673859600308595000],
    close=[506.0, 505.0, 506.0, 506.0],
    volume=[3340, 1, 17, 2],
    bid_price=[505.0, 505.0, 506.0, 506.0],
    bid_volume=[122, 320, 60, 58],
    ask_price=[506.0, 506.0, 507.0, 507.0],
    ask_volume=[13, 22, 702, 702]
    tick_type=[1, 2, 1, 2]
)
```

TICKS

In

```
ticks = api.ticks(
    contract=api.Contracts.Stocks["2330"],
    date="2023-01-16",
    query_type=sj.constant.TicksQueryType.LastCount,
    last_cnt=4,
)
ticks
```

Out

```
Ticks(
    ts=[1673859600113699000, 1673859600228800000, 1673859600244294000, 1673859600308595000],
    close=[506.0, 505.0, 506.0, 506.0],
    volume=[3340, 1, 17, 2],
    bid_price=[505.0, 505.0, 506.0, 506.0],
    bid_volume=[122, 320, 60, 58],
    ask_price=[506.0, 506.0, 507.0, 507.0],
    ask_volume=[13, 22, 702, 702]
    tick_type=[1, 2, 1, 2]
)
```

### KBar

**Kbars**

```
api.kbars?

Signature:
api.kbars(
    contract: shioaji.contracts.BaseContract,
    start: str = '2023-01-15',
    end: str = '2023-01-16',
    timeout: int = 30000,
    cb: Callable[[shioaji.data.Kbars], NoneType] = None,
) -> shioaji.data.Kbars
Docstring:
get Kbar
```

**In**

```
kbars = api.kbars(
    contract=api.Contracts.Stocks["2330"],
    start="2023-01-15",
    end="2023-01-16",
)
kbars
```

**Out**

```
Kbars(
    ts=[1673859660000000000, 1673859720000000000, 1673859780000000000, 1673859840000000000],
    Open=[506.0, 505.0, 505.0, 504.0],
    High=[508.0, 506.0, 506.0, 505.0],
    Low=[505.0, 505.0, 504.0, 504.0],
    Close=[505.0, 505.0, 504.0, 504.0],
    Volume=[5308, 1018, 543, 209]
)
```

**Kbars**

```
ts (int): timestamp
Open (float): open price
High (float): the highest price
Low: (float): the lowest price
Close (float): close price
Volume (int): volume
```

DataFrame

**In**

```
import pandas as pd
df = pd.DataFrame({**kbars})
df.ts = pd.to_datetime(df.ts)
df.head()
```

**Out**

| Close | Amount | Low | Volume | ts | Open | High |
|---|---|---|---|---|---|---|
| 505 | 2.68731e+09 | 505 | 5308 | 2023-01-16 09:01:00 | 506 | 508 |
| 505 | 5.14132e+08 | 505 | 1018 | 2023-01-16 09:02:00 | 505 | 506 |
| 504 | 2.74112e+08 | 504 | 543 | 2023-01-16 09:03:00 | 505 | 506 |
| 504 | 1.0542e+08 | 504 | 209 | 2023-01-16 09:04:00 | 504 | 505 |

**Historical Periods**

| | Start Date | End Date |
|---|---|---|
| Index | 2020-03-02 | Today |
| Stock | 2020-03-02 | Today |
| Futures | 2020-03-22 | Today |

`api.Contracts` `R1, R2` `R1, R2`
`api.Contracts.Futures.TXF.TXFR1` `R1, R2` `Ticks` `Kbars`

Ticks

**Ticks**

```
ticks = api.ticks(
    contract=api.Contracts.Futures.TXF.TXFR1,
    date="2020-03-22"
)
ticks
```

**Out**

```
Ticks(
    ts=[1616166000030000000, 1616166000140000000, 1616166000140000000, 1616166000190000000],
    close=[16011.0, 16013.0, 16014.0, 16011.0],
    volume=[49, 2, 2, 1],
    bid_price=[0.0, 16011.0, 16011.0, 16011.0],
    bid_volume=[0, 1, 1, 1],
    ask_price=[0.0, 16013.0, 16013.0, 16013.0],
    ask_volume=[0, 1, 1, 1]
    tick_type=[1, 1, 1, 2]
)
```

Kbars

### Kbars

```
kbars = api.kbars(
    contract=api.Contracts.Futures.TXF.TXFR1,
    start="2023-01-15",
    end="2023-01-16",
)
kbars
```

### Out

```
Kbars(
    ts=[1616402760000000000, 1616402820000000000, 1616402880000000000, 1616402940000000000],
    Open=[16018.0, 16018.0, 16000.0, 15992.0],
    High=[16022.0, 16020.0, 16005.0, 15999.0],
    Low=[16004.0, 16000.0, 15975.0, 15989.0],
    Close=[16019.0, 16002.0, 15992.0, 15994.0],
    Volume=[1791, 864, 1183, 342]
)
```

## 4.4.3

500

```
>> api.snapshots?

Signature:
api.snapshots(
    contracts: List[Union[shioaji.contracts.Option, shioaji.contracts.Future, shioaji.contracts.Stock, shioaji.contracts.Index]],
    timeout: int = 30000,
    cb: Callable[[shioaji.data.Snapshot], NoneType] = None,
) -> List[shioaji.data.Snapshot]
Docstring:
get contract snapshot info
```

```
contracts = [api.Contracts.Stocks['2330'],api.Contracts.Stocks['2317']]
snapshots = api.snapshots(contracts)
snapshots
```

**Out**

```
[
    Snapshot(
        ts=1673620200000000000,
        code='2330',
        exchange='TSE',
        open=507.0,
        high=509.0,
        low=499.0,
        close=500.0,
        tick_type=<TickType.Sell: 'Sell'>,
        change_price=13.5,
        change_rate=2.77,
        change_type=<ChangeType.Up: 'Up'>,
        average_price=502.42,
        volume=48,
        total_volume=77606,
        amount=24000000,
        total_amount=38990557755,
        yesterday_volume=20963.0,
        buy_price=500.0,
        buy_volume=122.0,
        sell_price=501.0,
        sell_volume=1067,
        volume_ratio=3.7
    ),
    Snapshot(
        ts=1673620200000000000,
        code='2317',
        exchange='TSE',
        open=99.0,
        high=99.5,
        low=98.6,
        close=98.6,
        tick_type=<TickType.Sell: 'Sell'>,
        change_price=0.0,
        change_rate=0.0,
        change_type=<ChangeType.Unchanged: 'Unchanged'>,
        average_price=98.96,
        volume=63,
        total_volume=17809,
        amount=6211800,
        total_amount=1762344817,
        yesterday_volume=18537.0,
        buy_price=98.6,
        buy_volume=607.0,
        sell_price=98.7,
        sell_volume=4,
        volume_ratio=0.96
    )
]
```

Dataframe

**In**

```
df = pd.DataFrame(s.__dict__ for s in snapshots)
df.ts = pd.to_datetime(df.ts)
df
```

**Out**

| ts | code | exchange | open | high | low | close | tick_type | change_price | cl |
|---|---|---|---|---|---|---|---|---|---|
| 2023-01-13 14:30:00 | 2330 | TSE | 507 | 509 | 499 | 500 | Sell | 13.5 | |
| 2023-01-13 14:30:00 | 2317 | TSE | 99 | 99.5 | 98.6 | 98.6 | Sell | 0 | |

### Snapshot

```
ts (int):
code (str):
exchange (Exchange):
open (float):
high (float):
low (float):
close (float):
tick_type (TickType):       {None, Buy, Sell}
change_price (float):
change_rate (float):
change_type (ChangeType):    {LimitUp, Up, Unchanged, Down, LimitDown}
avgerage_price (float):
volume (int):
total_volume (int):
amount (int):
total_amount (int):
yestoday_volume (float):
buy_price (float):
buy_volume (float):
sell_price (float):
sell_volume (int):
volume_ratio (float):
```

## 4.4.4

```
>> api.short_stock_sources?

Signature:
api.short_stock_sources(
    contracts: List[shioaji.contracts.Stock],
    timeout: int = 5000,
    cb: Callable[[shioaji.data.ShortStockSource], NoneType] = None,
) -> List[shioaji.data.ShortStockSource]
```

**In**

```
contracts = [
    api.Contracts.Stocks['2330'],
    api.Contracts.Stocks['2317']
]
short_stock_sources = api.short_stock_sources(contracts)
short_stock_sources
```

**Out**

```
[
    ShortStockSource(code='2330', short_stock_source=58260, ts=1673943433000000000),
    ShortStockSource(code='2317', short_stock_source=75049, ts=1673943433000000000)
]
```

DataFrame

**In**

```
df = pd.DataFrame(s.__dict__ for s in short_stock_sources)
df.ts = pd.to_datetime(df.ts)
df
```

**Out**

| | code | short_stock_source | ts |
|---|---|---|---|
| | 2330 | 58260 | 2023-01-17 08:17:13 |
| | 2317 | 75049 | 2023-01-17 08:17:13 |

**ShortStockSource**

```
code (str):
short_stock_source (float):
ts (int):
```

## 4.4.5

### Credit Enquires

```
>> api.credit_enquires?

Signature:
api.credit_enquires(
    contracts: List[shioaji.contracts.Stock],
    timeout: int = 30000,
    cb: Callable[[shioaji.data.CreditEnquire], NoneType] = None,
) -> List[shioaji.data.CreditEnquire]
```

### In

```
contracts = [
    api.Contracts.Stocks['2330'],
    api.Contracts.Stocks['2890']
]
credit_enquires = api.credit_enquires(contracts)
credit_enquires
```

### Out

```
[
    CreditEnquire(update_time='2020-12-11 13:30:13', system='HE', stock_id='2330', margin_unit=1381),
    CreditEnquire(update_time='2020-12-11 13:30:02', system='HC', stock_id='2330', margin_unit=1371),
    CreditEnquire(update_time='2020-12-11 13:30:05', system='HN', stock_id='2330', margin_unit=1357),
    CreditEnquire(update_time='2020-12-11 13:30:03', system='HF', stock_id='2330', margin_unit=1314),
    CreditEnquire(update_time='2020-12-09 10:56:05', system='HE', stock_id='2890'),
    CreditEnquire(update_time='2020-12-11 09:33:04', system='HN', stock_id='2890'),
    CreditEnquire(update_time='2020-12-02 09:01:03', system='HF', stock_id='2890')
]
```

DataFrame

### In

```
df = pd.DataFrame(c.__dict__ for c in credit_enquires)
df.update_time = pd.to_datetime(df.update_time)
df
```

**Out**

| | margin_unit | short_unit | stock_id | system | update_time |
|---|---|---|---|---|---|
| 0 | 1381 | 0 | 2330 | HE | 2020-12-11 13:30:13 |
| 1 | 1371 | 0 | 2330 | HC | 2020-12-11 13:30:02 |
| 2 | 1357 | 0 | 2330 | HN | 2020-12-11 14:31:19 |
| 3 | 1314 | 0 | 2330 | HF | 2020-12-11 14:31:19 |
| 4 | 0 | 0 | 2890 | HE | 2020-12-09 10:56:05 |
| 5 | 0 | 0 | 2890 | HN | 2020-12-11 09:33:04 |
| 6 | 0 | 0 | 2890 | HF | 2020-12-02 09:01:03 |

**CreditEnquire**

```
update_time (str):
system (str):
stock_id (str):
margin_unit (int):
short_unit (int):
```

## 4.4.6

`Scanners` `scanner_type`

**Scanners**

```
>> api.scanners?

Signature:
api.scanners(
    scanner_type: shioaji.constant.ScannerType,
    ascending: bool = True,
    date: str = None,
    count: shioaji.shioaji.ConstrainedIntValue = 100, # 0 <= count <= 200
    timeout: int = 30000,
    cb: Callable[[List[shioaji.data.ChangePercentRank]], NoneType] = None,
) -> List[shioaji.data.ChangePercentRank]
```

`ascending` `True` `ascending` `False` `count`

```
ChangePercentRank:
ChangePriceRank:
DayRangeRank:
VolumeRank:
AmountRank:
```

```
scanners = api.scanners(
    scanner_type=sj.constant.ScannerType.ChangePercentRank,
    count=1
)
scanners
```

**Out**

```
[
    ChangePercentRank(
        date='2021-04-09',
        code='5211',
        name='  ',
        ts=1617978600000000000,
        open=16.4,
        high=17.6,
        low=16.35,
        close=17.6,
        price_range=1.25,
        tick_type=1,
        change_price=1.6,
        change_type=1,
        average_price=17.45,
        volume=7,
        total_volume=1742,
        amount=123200,
        total_amount=30397496,
        yesterday_volume=514,
        volume_ratio=3.39,
        buy_price=17.6,
        buy_volume=723,
        sell_price=0.0,
        sell_volume=0,
        bid_orders=237,
        bid_volumes=82,
        ask_orders=33,
        ask_volumes=64
    )
]
```

DataFrame

**In**

```
scanners = api.scanners(
    scanner_type=sj.constant.ScannerType.ChangePercentRank,
    count=5
)
df = pd.DataFrame(s.__dict__ for s in scanners)
df.ts = pd.to_datetime(df.ts)
df
```

**Out**

| date | code | name | ts | open | high | low | close | price_range |
|---|---|---|---|---|---|---|---|---|
| 2023-01-17 | 6259 | | 2023-01-17 11:11:41.030000 | 22.8 | 23.75 | 22.45 | 23.75 | 1.3 |
| 2023-01-17 | 6788 | | 2023-01-17 11:19:01.924000 | 107 | 116 | 107 | 116 | 9 |
| 2023-01-17 | 2540 | | 2023-01-17 11:17:39.435000 | 85.2 | 85.2 | 83 | 85.2 | 2.2 |
| 2023-01-17 | 8478 | | 2023-01-17 11:18:33.702000 | 350.5 | 378 | 347 | 378 | 31 |
| 2023-01-17 | 6612 | | 2023-01-17 11:15:32.752000 | 102 | 109 | 102 | 109 | 7 |

**ChangePercentRank**

```
date (str):
code (str):
name (str):
ts (int):
open (float):
high (float):
low (float):
close (float):
price_range (float):     (   -   )
tick_type (int):      {1:   , 2:   , 0:     }
change_price (float):
change_type (int):
    {LimitUp, Up, Unchanged, Down, LimitDown}
average_price (float):
volume (int):
total_volume (int):
amount (int):
total_amount (int):
yesterday_volume (int):
volume_ratio (float):     /
buy_price (float):
buy_volume (int):
sell_price (float):
sell_volume (int):
bid_orders (int):
bid_volumes (int):
ask_orders (int):
ask_volumes (int):
```

# 4.5

## 4.5.1

**version>=1.0** **version<1.0**

```
price (float or int):
quantity (int):
action (str): {Buy:  , Sell:  }
price_type (str): {LMT:   , MKT:   , MKP:     }
order_type (str):      {ROD, IOC, FOK}
order_cond (str): {Cash:  , MarginTrading:  , ShortSelling:  }
order_lot (str): {
        Common:  ,
        Fixing:  ,
        Odd:    ,
        IntradayOdd:
    }
daytrade_short (bool):
custom_field (str):                          6
account (:obj:Account):
ca (binary):

price (float or int):
quantity (int):
action (str): {Buy:  , Sell:  }
price_type (str): {LMT:   , MKT:   , MKP:     }
order_type (str):      {ROD, IOC, FOK}
order_cond (str): {Cash:  , MarginTrading:  , ShortSelling:  }
order_lot (str): {
        Common:  ,
        Fixing:  ,
        Odd:    ,
        IntradayOdd:
    }
first_sell (str):      {true, false}
custom_field (str):                          6
account (:obj:Account):
ca (binary):
```

`contract` `order`

```
api.place_order?

    Signature:
        api.place_order(
            contract: shioaji.contracts.Contract,
            order: shioaji.order.Order,
            timeout: int = 5000,
            cb: Callable[[shioaji.order.Trade], NoneType] = None,
        ) -> shioaji.order.Trade
    Docstring:
    placing order
```

```
contract = api.Contracts.Stocks.TSE.TSE2890
```

**version>=1.0** **version<1.0**

```
order = api.Order(
    price=17,
    quantity=3,
    action=sj.constant.Action.Buy,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    order_lot=sj.constant.StockOrderLot.Common,
    # daytrade_short=False,
    custom_field="test",
    account=api.stock_account
)

order = api.Order(
    price=17,
    quantity=3,
    action=sj.constant.Action.Buy,
    price_type=sj.constant.TFTStockPriceType.LMT,
    order_type=sj.constant.TFTOrderType.ROD,
    order_lot=sj.constant.TFTStockOrderLot.Common,
    # first_sell=sj.constant.StockFirstSell.No,
    custom_field="test",
    account=api.stock_account
)
```

```
trade = api.place_order(contract, order)
trade
```

**Out**

```
Trade(
    contract=Stock(
        exchange=<Exchange.TSE: 'TSE'>,
        code='2890',
        symbol='TSE2890',
        name='   ',
        category='17',
        unit=1000,
        limit_up=19.05,
        limit_down=15.65,
        reference=17.35,
        update_date='2023/01/12',
        day_trade=<DayTrade.Yes: 'Yes'>
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=17,
        quantity=3,
        id='531e27af',
        seqno='000002',
        ordno='000001',
        account=Account(
            account_type=<AccountType.Stock: 'S'>,
            person_id='A123456789',
            broker_id='9A95',
            account_id='1234567',
            signed=True
        ),
        custom_field='test',
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>,
        daytrade_short=False
    ),
    status=OrderStatus(
        id='531e27af',
        status=<Status.PendingSubmit: 'PendingSubmit'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 11, 18, 3, 867490),
        deals=[]
    )
)
```

`update_status` `trade`

( )

```
api.update_status(api.stock_account)
trade
```

Out

```
Trade(
    contract=Stock(
        exchange=<Exchange.TSE: 'TSE'>,
        code='2890',
        symbol='TSE2890',
        name='    ',
        category='17',
        unit=1000,
        limit_up=19.05,
        limit_down=15.65,
        reference=17.35,
        update_date='2023/01/12',
        day_trade=<DayTrade.Yes: 'Yes'>
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=17,
        quantity=3,
        id='531e27af',
        seqno='000002',
        ordno='000001',
        account=Account(
            account_type=<AccountType.Stock: 'S'>,
            person_id='A123456789',
            broker_id='9A95',
            account_id='1234567',
            signed=True
        ),
        custom_field='test',
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>,
        daytrade_short=False
    ),
    status=OrderStatus(
        id='531e27af',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 11, 18, 3, 867490),
        order_quantity=3,
        deals=[]
    )
)
```

- `PendingSubmit`:
- `PreSubmitted`:
- `Submitted`:
- `Failed`:
- `Cancelled`:
- `Filled`:
- `Filling`:

```
api.update_order?

    Signature:
        api.update_order(
            trade: shioaji.order.Trade,
            price: Union[pydantic.types.StrictInt, float] = None,
            qty: int = None,
            timeout: int = 5000,
            cb: Ca  lable[[shioaji.order.Trade], NoneType] = None,
        ) -> shioaji.order.Trade
    Docstring: update the order price or qty
```

```
api.update_order(trade=trade, price=17.5)
api.update_status(api.stock_account)
trade
```

**Out**

```
Trade(
    contract=Stock(
        exchange=<Exchange.TSE: 'TSE'>,
        code='2890',
        symbol='TSE2890',
        name='    ',
        category='17',
        unit=1000,
        limit_up=19.05,
        limit_down=15.65,
        reference=17.35,
        update_date='2023/01/12',
        day_trade=<DayTrade.Yes: 'Yes'>
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=17,
        quantity=3,
        id='531e27af',
        seqno='000002',
        ordno='000001',
        account=Account(
            account_type=<AccountType.Stock: 'S'>,
            person_id='A123456789',
            broker_id='9A95',
            account_id='1234567',
            signed=True
        ),
        custom_field='test',
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>,
        daytrade_short=False
    ),
    status=OrderStatus(
        id='531e27af',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 11, 18, 3, 867490),
        modified_price=17.5,
        order_quantity=3,
        deals=[]
    )
)
```

( )

`update_order`

( )

```
api.update_order(trade=trade, qty=1)
api.update_status(api.stock_account)
trade
```

Out

```
Trade(
    contract=Stock(
        exchange=<Exchange.TSE: 'TSE'>,
        code='2890',
        symbol='TSE2890',
        name='    ',
        category='17',
        unit=1000,
        limit_up=19.05,
        limit_down=15.65,
        reference=17.35,
        update_date='2023/01/12',
        day_trade=<DayTrade.Yes: 'Yes'>
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=17,
        quantity=3,
        id='531e27af',
        seqno='000002',
        ordno='000001',
        account=Account(
            account_type=<AccountType.Stock: 'S'>,
            person_id='A123456789',
            broker_id='9A95',
            account_id='1234567',
            signed=True
        ),
        custom_field='test',
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>,
        daytrade_short=False
    ),
    status=OrderStatus(
        id='531e27af',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 11, 18, 3, 867490),
        order_quantity=3,
        cancel_quantity=1,
        deals=[]
    )
)
```

```
api.cancel_order(trade)
api.update_status(api.stock_account)
trade
```

**Out**

```
Trade(
    contract=Stock(
        exchange=<Exchange.TSE: 'TSE'>,
        code='2890',
        symbol='TSE2890',
        name='   ',
        category='17',
        unit=1000,
        limit_up=19.05,
        limit_down=15.65,
        reference=17.35,
        update_date='2023/01/12',
        day_trade=<DayTrade.Yes: 'Yes'>
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=17,
        quantity=3,
        id='531e27af',
        seqno='000002',
        ordno='000001',
        account=Account(
            account_type=<AccountType.Stock: 'S'>,
            person_id='A123456789',
            broker_id='9A95',
            account_id='1234567',
            signed=True
        ),
        custom_field='test',
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>,
        daytrade_short=False
    ),
    status=OrderStatus(
        id='531e27af',
        status=<Status.Cancelled: 'Cancelled'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 11, 18, 3, 867490),
        order_quantity=3,
        cancel_quantity=3,
        deals=[]
    )
)
```

```
api.update_status(api.stock_account)
trade
```

**Out**

```
Trade(
    contract=Stock(
        exchange=<Exchange.TSE: 'TSE'>,
        code='2890',
        symbol='TSE2890',
        name='   ',
        category='17',
        unit=1000,
        limit_up=19.05,
        limit_down=15.65,
        reference=17.35,
        update_date='2023/01/12',
        day_trade=<DayTrade.Yes: 'Yes'>
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=17,
        quantity=3,
        id='531e27af',
        seqno='000002',
        ordno='000001',
        account=Account(
            account_type=<AccountType.Stock: 'S'>,
            person_id='A123456789',
            broker_id='9A95',
            account_id='1234567',
            signed=True
        ),
        custom_field='test',
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>,
        daytrade_short=False
    ),
    status=OrderStatus(
        id='531e27af',
        status=<Status.Filled: 'Filled'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 11, 18, 3, 867490),
        order_quantity=3,
        deals=[
            Deal(seq='000001', price=17, quantity=3, ts=1673501631.62918)
        ]
    )
)
```

( jupyter)

```
order = api.Order(
    price=12,
    quantity=1,
    action=sj.constant.Action.Buy,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    order_lot=sj.constant.StockOrderLot.Common,
    custom_field="test",
    account=api.stock_account
)
```

```
order = api.Order(
    price=12,
    quantity=1,
    action=sj.constant.Action.Sell,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    order_lot=sj.constant.StockOrderLot.Common,
    custom_field="test",
    account=api.stock_account
)
```

**Daytrade Short**

version>=1.0 version<1.0

```
order = api.Order(
    price=12,
    quantity=1,
    action=sj.constant.Action.Sell,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    order_lot=sj.constant.StockOrderLot.Common,
    daytrade_short=True,
    custom_field="test",
    account=api.stock_account
)

order = api.Order(
    price=12,
    quantity=1,
    action=sj.constant.Action.Sell,
    price_type=sj.constant.TFTStockPriceType.LMT,
    order_type=sj.constant.TFTOrderType.ROD,
    order_lot=sj.constant.TFTStockOrderLot.Common,
    first_sell=sj.constant.StockFirstSell.Yes,
    custom_field="test",
    account=api.stock_account
)
```

**ROD + LMT**

**ROD + LMT**

version>=1.0 version<1.0

```
order = api.Order(
    price=12,
    quantity=1,
    action=sj.constant.Action.Sell,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    order_lot=sj.constant.StockOrderLot.Common,
    custom_field="test",
    account=api.stock_account
)

order = api.Order(
    price=12,
    quantity=1,
    action=sj.constant.Action.Sell,
    price_type=sj.constant.TFTStockPriceType.LMT,
    order_type=sj.constant.TFTOrderType.ROD,
    order_lot=sj.constant.TFTStockOrderLot.Common,
    custom_field="test",
    account=api.stock_account
)
```

## 4.5.2

```
price (float or int):
quantity (int):
action (str): {Buy:  , Sell:  }
price_type (str): {LMT:   , MKT:   , MKP:    }
order_type (str):      {ROD, IOC, FOK}
octype (str): {Auto:   , New:   , Cover:   , DayTrade:   }
account (:obj:Account):
ca (binary):
```

`contract` `order`

```
api.place_order?

    Signature:
        api.place_order(
            contract: shioaji.contracts.Contract,
            order: shioaji.order.Order,
            timeout: int = 5000,
            cb: Callable[[shioaji.order.Trade], NoneType] = None,
        ) -> shioaji.order.Trade
    Docstring:
    placing order
```

```
contract = api.Contracts.Futures.TXF.TXF202301
```

**version>=1.0** **version<1.0**

```
order = api.Order(
    action=sj.constant.Action.Buy,
    price=14400,
    quantity=3,
    price_type=sj.constant.FuturesPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    octype=sj.constant.FuturesOCType.Auto,
    account=api.futopt_account
)

order = api.Order(
    action=sj.constant.Action.Buy,
    price=14400,
    quantity=3,
    price_type=sj.constant.FuturesPriceType.LMT,
    order_type=sj.constant.FuturesOrderType.ROD,
    octype=sj.constant.FuturesOCType.Auto,
    account=api.futopt_account
)
```

```
trade = api.place_order(contract, order)
trade
```

**Out**

```
Trade(
    contract=Future(
        code='TXFA3',
        symbol='TXF202301',
        name='    01',
        category='TXF',
        delivery_month='202301',
        delivery_date='2023/01/30',
        underlying_kind='I',
        unit=1,
        limit_up=16270.0,
        limit_down=13312.0,
        reference=14791.0,
        update_date='2023/01/12'
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=14400,
        quantity=3,
        id='5efffde1',
        seqno='000004',
        ordno='000003',
        account=Account(
            account_type=<AccountType.Future: 'F'>,
            person_id='A123456789',
            broker_id='F002000',
            account_id='1234567',
            signed=True
        ),
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>
    ),
    status=OrderStatus(
        id='5efffde1',
        status=<Status.PendingSubmit: 'PendingSubmit'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 14, 56, 13, 995651),
        deals=[]
    )
)
```

`update_status` `trade`

```
api.update_status(api.futopt_account)
trade
```

**Out**

```
Trade(
    contract=Future(
        code='TXFA3',
        symbol='TXF202301',
        name='    01',
        category='TXF',
        delivery_month='202301',
        delivery_date='2023/01/30',
        underlying_kind='I',
        unit=1,
        limit_up=16270.0,
        limit_down=13312.0,
        reference=14791.0,
        update_date='2023/01/12'
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=14400,
        quantity=3,
        id='5efffde1',
        seqno='000004',
        ordno='000003',
        account=Account(
            account_type=<AccountType.Future: 'F'>,
            person_id='A123456789',
            broker_id='F002000',
            account_id='1234567',
            signed=True
        ),
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>
    ),
    status=OrderStatus(
        id='5efffde1',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 14, 56, 13, 995651),
        deals=[]
    )
)
```

- `PendingSubmit`:
- `PreSubmitted`:
- `Submitted`:
- `Failed`:
- `Cancelled`:
- `Filled`:
- `Filling`:

```
api.update_order?

    Signature:
        api.update_order(
            trade: shioaji.order.Trade,
            price: Union[pydantic.types.StrictInt, float] = None,
            qty: int = None,
            timeout: int = 5000,
            cb: Ca  lable[[shioaji.order.Trade], NoneType] = None,
        ) -> shioaji.order.Trade
    Docstring: update the order price or qty
```

```
api.update_order(trade=trade, price=14450)
api.update_status(api.futopt_account)
trade
```

**Out**

```
Trade(
    contract=Future(
        code='TXFA3',
        symbol='TXF202301',
        name='    01',
        category='TXF',
        delivery_month='202301',
        delivery_date='2023/01/30',
        underlying_kind='I',
        unit=1,
        limit_up=16270.0,
        limit_down=13312.0,
        reference=14791.0,
        update_date='2023/01/12'
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=14400,
        quantity=3,
        id='5efffde1',
        seqno='000004',
        ordno='000003',
        account=Account(
            account_type=<AccountType.Future: 'F'>,
            person_id='A123456789',
            broker_id='F002000',
            account_id='1234567',
            signed=True
        ),
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>
    ),
    status=OrderStatus(
        id='5efffde1',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 14, 56, 13, 995651),
        modified_price=14450,
        order_quantity=3,
        deals=[]
    )
)
```

( )

`update_order`

**( )**

```
api.update_order(trade=trade, qty=1)
api.update_status(api.futopt_account)
trade
```

**Out**

```
Trade(
    contract=Future(
        code='TXFA3',
        symbol='TXF202301',
        name='    01',
        category='TXF',
        delivery_month='202301',
        delivery_date='2023/01/30',
        underlying_kind='I',
        unit=1,
        limit_up=16270.0,
        limit_down=13312.0,
        reference=14791.0,
        update_date='2023/01/12'
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=14400,
        quantity=3,
        id='5efffde1',
        seqno='000004',
        ordno='000003',
        account=Account(
            account_type=<AccountType.Future: 'F'>,
            person_id='A123456789',
            broker_id='F002000',
            account_id='1234567',
            signed=True
        ),
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>
    ),
    status=OrderStatus(
        id='5efffde1',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 14, 56, 13, 995651),
        order_quantity=3,
        cancel_quantity=1,
        deals=[]
    )
)
```

```
api.cancel_order(trade)
api.update_status(api.futopt_account)
trade
```

**Out**

```
Trade(
    contract=Future(
        code='TXFA3',
        symbol='TXF202301',
        name='    01',
        category='TXF',
        delivery_month='202301',
        delivery_date='2023/01/30',
        underlying_kind='I',
        unit=1,
        limit_up=16270.0,
        limit_down=13312.0,
        reference=14791.0,
        update_date='2023/01/12'
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=14400,
        quantity=3,
        id='5efffde1',
        seqno='000004',
        ordno='000003',
        account=Account(
            account_type=<AccountType.Future: 'F'>,
            person_id='A123456789',
            broker_id='F002000',
            account_id='1234567',
            signed=True
        ),
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>
    ),
    status=OrderStatus(
        id='5efffde1',
        status=<Status.Cancelled: 'Cancelled'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 14, 56, 13, 995651),
        order_quantity=3,
        cancel_quantity=3,
        deals=[]
    )
)
```

**( )**

```
api.update_status(api.futopt_account)
trade
```

**Out**

```
Trade(
    contract=Future(
        code='TXFA3',
        symbol='TXF202301',
        name='    01',
        category='TXF',
        delivery_month='202301',
        delivery_date='2023/01/30',
        underlying_kind='I',
        unit=1,
        limit_up=16270.0,
        limit_down=13312.0,
        reference=14791.0,
        update_date='2023/01/12'
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=14400,
        quantity=3,
        id='5efffde1',
        seqno='000004',
        ordno='000003',
        account=Account(
            account_type=<AccountType.Future: 'F'>,
            person_id='A123456789',
            broker_id='F002000',
            account_id='1234567',
            signed=True
        ),
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>
    ),
    status=OrderStatus(
        id='5efffde1',
        status=<Status.Filled: 'Filled'>,
        status_code='00',
        order_datetime=datetime.datetime(2023, 1, 12, 14, 56, 13, 995651),
        order_quantity=3,
        deals=[
            Deal(seq='000001', price=14400, quantity=3, ts=1673501631.62918)
        ]
    )
)
```

( jupyter)

```
order = api.Order(
    action=sj.constant.Action.Buy,
    price=14400,
    quantity=2,
    price_type=sj.constant.FuturesPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    octype=sj.constant.FuturesOCType.Auto,
    account=api.futopt_account
)
```

```
order = api.Order(
    action=sj.constant.Action.Sell,
    price=14400,
    quantity=2,
    price_type=sj.constant.FuturesPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    octype=sj.constant.FuturesOCType.Auto,
    account=api.futopt_account
)
```

**ROD + LMT**

### ROD + LMT

```
order = api.Order(
    action=sj.constant.Action.Sell,
    price=14400,
    quantity=2,
    price_type=sj.constant.FuturesPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    octype=sj.constant.FuturesOCType.Auto,
    account=api.futopt_account
)
```

## 4.5.3

( jupyter)

```
contract = api.Contracts.Stocks.TSE.TSE0050
order = api.Order(
    price=90,
    quantity=10,
    action=sj.constant.Action.Buy,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    order_lot=sj.constant.StockOrderLot.IntradayOdd,
    account=api.stock_account,
)

trade = api.place_order(contract, order)
trade
```

**Out**

```
Trade(
    contract=Stock(
        exchange=<Exchange.TSE: 'TSE'>,
        code='0050',
        symbol='TSE0050',
        name='    50',
        category='00',
        limit_up=115.8,
        limit_down=94.8,
        eference=105.3,
        update_date='2020/09/21',
        margin_trading_balance=15390,
        short_selling_balance=2,
        day_trade=<DayTrade.Yes: 'Yes'>
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=90.0,
        quantity=10,
        id='38e68afe',
        seqno='482283',
        ordno='WA313',
        account=Account(
            account_type=<AccountType.Stock: 'S'>,
            person_id='YOUR_PERSON_ID',
            broker_id='9A95',
            account_id='0506112',
            signed=True
        ),
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>,
        order_lot=<StockOrderLot.IntradayOdd: 'IntradayOdd'>
    ),
    status=OrderStatus(
        id='38e68afe',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime.datetime(2020, 9, 21, 14, 38, 51),
        deals=[]
    )
)
```

`update_order`

( )

```
api.update_order(trade=trade, qty=2)
api.update_status(api.stock_account)
trade
```

Out

```
Trade(
    contract=Stock(
        exchange=<Exchange.TSE: 'TSE'>,
        code='0050',
        symbol='TSE0050',
        name='    50',
        category='00',
        limit_up=115.8,
        limit_down=94.8,
        reference=105.3,
        update_date='2020/09/21',
        margin_trading_balance=15390,
        short_selling_balance=2,
        day_trade=<DayTrade.Yes: 'Yes'>
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=90.0,
        quantity=10,
        id='9b44c3b2',
        seqno='482293',
        ordno='WA328',
        account=Account(
            account_type=<AccountType.Stock: 'S'>,
            person_id='YOUR_PERSON_ID',
            broker_id='9A95',
            account_id='0506112',
            signed=True
        ),
    price_type=<StockPriceType.LMT: 'LMT'>,
    order_type=<OrderType.ROD: 'ROD'>,
    order_lot=<StockOrderLot.IntradayOdd: 'IntradayOdd'>),
    status=OrderStatus(
        id='9b44c3b2',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime.datetime(2020, 9, 21, 14, 54, 36),
        cancel_quantity=2,
        deals=[]
    )
)
```

```
api.cancel_order(trade)
api.update_status(api.stock_account)
trade
```

**Out**

```
Trade(
    contract=Stock(
        exchange=<Exchange.TSE: 'TSE'>,
        code='0050',
        symbol='TSE0050',
        name='    50',
        category='00',
        limit_up=115.8,
        limit_down=94.8,
        reference=105.3,
        update_date='2020/09/21',
        margin_trading_balance=15390,
        short_selling_balance=2,
        day_trade=<DayTrade.Yes: 'Yes'>
    ),
    order=Order(
        action=<Action.Buy: 'Buy'>,
        price=90.0,
        quantity=10,
        id='9b44c3b2',
        seqno='482293',
        ordno='WA328',
        account=Account(
            account_type=<AccountType.Stock: 'S'>,
            person_id='YOUR_PERSON_ID',
            broker_id='9A95',
            account_id='0506112',
            signed=True
        ),
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>,
        order_lot=<StockOrderLot.IntradayOdd: 'IntradayOdd'>
    ),
    status=OrderStatus(
        id='9b44c3b2',
        status=<Status.Cancelled: 'Cancelled'>,
        status_code='00',
        order_datetime=datetime.datetime(2020, 9, 21, 14, 54, 36),
        cancel_quantity=10,
        deals=[]
    )
)
```

## 4.5.4

:

**place_comboorder**

```
api.place_comboorder?

    Signature:
        api.place_comboorder(
            combo_contract: shioaji.contracts.ComboContract,
            order: shioaji.order.ComboOrder,
            timeout: int = 5000,
            cb: Callable[[shioaji.order.ComboTrade], NoneType] = None,
        )
    Docstring:
        placing combo order
```

`contract` `order`

```
contract_1 = api.Contracts.Options.TX4.TX4202111017850C
contract_2 = api.Contracts.Options.TX4.TX4202111017850P
combo_contract = sj.contracts.ComboContract(
    legs=[
        sj.contracts.ComboBase(action="Sell", **contract_1.dict()),
        sj.contracts.ComboBase(action="Sell", **contract_2.dict()),
    ]
)
```

```
order = api.ComboOrder(
    price_type="LMT",
    price=1,
    quantity=1,
    order_type="IOC",
    octype=sj.constant.FuturesOCType.New,
)
```

```
trade = api.place_comboorder(combo_contract, order)
```

`trade`

```
api.cancel_comboorder(trade)
```

`list_trades` `update_status` `update_combostatus`

In

```
api.update_combostatus()
api.list_combotrades()
```

Out

```
[
    ComboTrade(
        contract=ComboContract(
            legs=[
                ComboBase(
                    security_type=<SecurityType.Option: 'OPT'>,
                    exchange=<Exchange.TAIFEX: 'TAIFEX'>,
                    code='TX516000L1',
                    symbol='TX5202112016000C',
                    name='     12W5  16000C',
                    category='TX5',
                    delivery_month='202112',
                    delivery_date='2021/12/29',
                    strike_price=16000.0,
                    option_right=<OptionRight.Call: 'C'>,
                    underlying_kind='I',
                    unit=1,
                    limit_up=3630.0,
                    limit_down=68.0,
                    reference=1850.0,
                    update_date='2021/12/23',
                    action=<Action.Sell: 'Sell'>),
                ComboBase(
                    security_type=<SecurityType.Option: 'OPT'>,
                    exchange=<Exchange.TAIFEX: 'TAIFEX'>,
                    code='TX516000X1',
                    symbol='TX5202112016000P',
                    name='     12W5  16000P',
                    category='TX5',
                    delivery_month='202112',
                    delivery_date='2021/12/29',
                    strike_price=16000.0,
                    option_right=<OptionRight.Put: 'P'>,
                    underlying_kind='I',
                    unit=1,
                    limit_up=1780.0,
                    limit_down=0.1,
                    reference=0.9,
                    update_date='2021/12/23',
                    action=<Action.Sell: 'Sell'>)
                ]
            ),
        order=Order(
            action=<Action.Sell: 'Sell'>,
            price=1.0,
            quantity=1,
            id='46989de8',
            seqno='743595',
            ordno='000000',
            account=Account(
                account_type=<AccountType.Future: 'F'>,
                person_id='YOUR_PERSON_ID',
                broker_id='F002000',
                account_id='1234567',
                signed=True
            ),
            price_type=<StockPriceType.LMT: 'LMT'>,
            order_type=<OrderType.IOC: 'IOC'>,
            octype=<FuturesOCType.New: 'New'>
        ),
        status=ComboStatus(
            id='46989de8',
            status=<Status.Failed: 'Failed'>,
            status_code='9909',
            order_datetime=datetime.datetime(2021, 12, 23, 8, 46, 47),
            msg='       ',
            modified_price=1.0,
            deals={}
        )
    )
]
```

## 4.5.5

:

- 
- 8:00~14:30

In

```
reserve_summary_resp = api.stock_reserve_summary(account)
```

Out

```
ReserveStocksSummaryResponse(
    response=ReserveStocksSummary(
        stocks=[
            ReserveStockSummary(
                contract=Contract(
                    security_type=<SecurityType.Stock: 'STK'>,
                    exchange=<Exchange.TSE: 'TSE'>,
                    code='2890',
                    name='    '
                ),
                available_share=5000,
                reserved_share=0
            )
        ],
        account=StockAccount(
            person_id='X123456789',
            broker_id='9A95',
            account_id='12345678',
            signed=True
            )
        )
    )
```

In

```
contract = api.Contracts.Stocks["2890"]
resp = api.reserve_stock(account, contract, 1000)
```

**Out**

```
ReserveStockResponse(
    response=ReserveOrderResp(
        contract=Stock(
            exchange=<Exchange.TSE: 'TSE'>,
            code='2890',
            symbol='TSE2890',
            name='    '
        ),
        account=StockAccount(
            person_id='X123456789',
            broker_id='9A95',
            account_id='12345678',
            signed=True),
        share=1000,
        status=True,
        info=''
    )
)
```

**In**

```
resp = api.stock_reserve_detail(account)
```

**Out**

```
ReserveStocksDetailResponse(
    response=ReserveStocksDetail(stocks=[
            ReserveStockDetail(
                contract=Contract(
                    security_type=<SecurityType.Stock: 'STK'>,
                    exchange=<Exchange.TSE: 'TSE'>,
                    code='6153',
                    name='    '
                ),
                share=1000,
                order_ts=1638253253,
                status=True,
                info='    '
            )
        ],
        account=StockAccount(
            person_id='X123456789',
            broker_id='9A95',
            account_id='12345678',
            signed=True)
        )
)
```

**In**

```
contract = api.Contracts.Stocks["2890"]
resp = api.reserve_earmarking(account, contract, 1000, 15.15)
```

**Out**

```
ReserveEarmarkingResponse(
    response=EarmarkingOrderResp(
        contract=Stock(
            exchange=<Exchange.TSE: 'TSE'>,
            code='2890',
            symbol='TSE2890',
            name='    ',
        ),
        account=StockAccount(
            person_id='X123456789',
            broker_id='9A95',
            account_id='12345678',
            signed=True)
        ),
        share=1000,
        price=15.15,
        status=True,
        info='OK')
)
```

**In**

```
api.earmarking_detail(account)
```

**Out**

```
EarmarkStocksDetailResponse(
    response=EarmarkStocksDetail(
        stocks=[
            EarmarkStockDetail(
                contract=Contract(
                    security_type=<SecurityType.Stock: 'STK'>,
                    exchange=<Exchange.TSE: 'TSE'>,
                    code='2890',
                    name='    '
                ),
                share=1000,
                price=15.15,
                amount=15171,
                order_ts=1638416488,
                status=False,
                info='    '),
            EarmarkStockDetail(
                contract=Contract(
                    security_type=<SecurityType.Stock: 'STK'>,
                    exchange=<Exchange.TSE: 'TSE'>,
                    code='2890',
                    name='    '
                ),
                share=1000,
                price=15.15,
                amount=15171,
                order_ts=1638415662, status=True,
                info='')
            ],
            account=StockAccount(
                person_id='X123456789',
                broker_id='9A95',
                account_id='12345678',
                signed=True)
            )
        )
    )
```

```python
import shioaji as sj

api = sj.Shioaji()
accounts = api.login("YOUR_PERSON_ID", "YOUR_PASSWORD", contracts_timeout=10000)
api.activate_ca(
    ca_path="/c/your/ca/path/Sinopac.pfx",
    ca_passwd="YOUR_CA_PASSWORD",
    person_id="Person of this Ca",
)
for account in accounts:
    if account.account_type == AccountType.Stock:
        reserve_summary_resp = api.stock_reserve_summary(account)
        for reserve_stock_summary in reserve_summary_resp.response.stocks:
                if reserve_stock_summary.available_share:
                    resp = api.reserve_stock(
                        account,
                        reserve_stock_summary.contract,
                        reserve_stock_summary.available_share
                    )
```

## 4.5.6

`Trade` `update_status` `trade` `trade` `OrderStatus` `update_status` `account`

**Update Status**

```
api.update_status?
```

**Out**

```
Signature:
    api.update_status(
        account: shioaji.account.Account = None,
        trade: shioaji.order.Trade = None,
        timeout: int = 5000,
        cb: Callable[[List[shioaji.order.Trade]], NoneType] = None,
    )
Docstring: update status of all trades you have
```

```
api.update_status(api.stock_account)
api.list_trades()
```

**Out**

```
[
    Trade(
        contract=Stock(
            exchange=<Exchange.TSE: 'TSE'>,
            code='2890',
            symbol='TSE2890',
            name='    ',
            category='17',
            unit=1000,
            limit_up=19.05,
            limit_down=15.65,
            reference=17.35,
            update_date='2023/01/12',
            day_trade=<DayTrade.Yes: 'Yes'>
        ),
        order=Order(
            action=<Action.Buy: 'Buy'>,
            price=17,
            quantity=3,
            id='531e27af',
            seqno='000002',
            ordno='000001',
            account=Account(
                account_type=<AccountType.Stock: 'S'>,
                person_id='A123456789',
                broker_id='9A95',
                account_id='1234567',
                signed=True
            ),
            custom_field='test',
            price_type=<StockPriceType.LMT: 'LMT'>,
            order_type=<OrderType.ROD: 'ROD'>,
            daytrade_short=True
        ),
        status=OrderStatus(
            id='531e27af',
            status=<Status.Filled: 'Filled'>,
            status_code='00',
            order_datetime=datetime.datetime(2023, 1, 12, 11, 18, 3, 867490),
            order_quantity=3,
            deals=[
                Deal(seq='000001', price=17, quantity=3, ts=1673501631.62918)
            ]
        )
    )
]
```

```
api.update_status(api.futopt_account)
api.list_trades()
```

**Out**

```
[
    Trade(
        contract=Future(
            code='TXFA3',
            symbol='TXF202301',
            name='    01',
            category='TXF',
            delivery_month='202301',
            delivery_date='2023/01/30',
            underlying_kind='I',
            unit=1,
            limit_up=16270.0,
            limit_down=13312.0,
            reference=14791.0,
            update_date='2023/01/12'
        ),
        order=Order(
            action=<Action.Buy: 'Buy'>,
            price=14400,
            quantity=3,
            id='5efffde1',
            seqno='000004',
            ordno='000003',
            account=Account(
                account_type=<AccountType.Future: 'F'>,
                person_id='A123456789',
                broker_id='F002000',
                account_id='1234567',
                signed=True
            ),
            price_type=<StockPriceType.LMT: 'LMT'>,
            order_type=<OrderType.ROD: 'ROD'>
        ),
        status=OrderStatus(
            id='5efffde1',
            status=<Status.Filled: 'Filled'>,
            status_code='00',
            order_datetime=datetime.datetime(2023, 1, 12, 14, 56, 13, 995651),
            order_quantity=3,
            deals=[
                Deal(seq='000001', price=14400, quantity=3, ts=1673501631.62918)
            ]
        )
    )
]
```

```
# you can get trade from place_order
# trade = api.place_order(contract, order)

# or get from api.list_trades
# trade = api.list_trades()[0]

api.update_status(trade=trade)
```

```
id (str):   Order
status (:obj:Status): {
            Cancelled:    ,
            Filled:     ,
            PartFilled:     ,
            Failed:   ,
            PendingSubmit:    ,
            PreSubmitted:    ,
            Submitted:
        }
status_code (str):
order_datetime (datetime):
order_quantity (int):
modified_price (float):
cancel_quantity (int):
deals (:List:Deal):
```

```
seq (str):
price (int or float):
quantity (int):
ts (float):
```

## 4.5.7

operation order status contract

**version>=1.0** **version<1.0**

```
OrderState.StockOrder {
    'operation': {
        'op_type': 'New',
        'op_code': '00',
        'op_msg': ''
    },
    'order': {
        'id': '97b63e2f',
        'seqno': '267677',
        'ordno': 'IM394',
        'account': {
            'account_type': 'S',
            'person_id': '',
            'broker_id': '9A95',
            'account_id': '1234567',
            'signed': True
        },
        'action': 'Buy',
        'price': 16.0,
        'quantity': 1,
        'order_type': 'ROD',
        'price_type': 'LMT',
        'order_cond': 'Cash',
        'order_lot': 'Common',
        'custom_field': 'test'
    },
    'status': {
        'id': '97b63e2f',
        'exchange_ts': 1673576134.038,
        'modified_price': 0.0,
        'cancel_quantity': 0,
        'order_quantity': 1,
        'web_id': '137'
    },
    'contract': {
        'security_type': 'STK',
        'exchange': 'TSE',
        'code': '2890',
        'symbol': '',
        'name': '',
        'currency': 'TWD'
    }
}
```

```
OrderState.TFTOrder {
    'operation': {
        'op_type': 'New',
        'op_code': '00',
        'op_msg': ''
    },
    'order': {
        'id': '97b63e2f',
        'seqno': '267677',
        'ordno': 'IM394',
        'account': {
            'account_type': 'S',
            'person_id': '',
            'broker_id': '9A95',
            'account_id': '1234567',
            'signed': True
        },
        'action': 'Buy',
        'price': 16.0,
        'quantity': 1,
        'order_type': 'ROD',
        'price_type': 'LMT',
        'order_cond': 'Cash',
        'order_lot': 'Common',
        'custom_field': 'test'
    },
    'status': {
        'id': '97b63e2f',
        'exchange_ts': 1673576134.038,
        'modified_price': 0.0,
        'cancel_quantity': 0,
        'order_quantity': 1,
        'web_id': '137'
    },
    'contract': {
        'security_type': 'STK',
        'exchange': 'TSE',
        'code': '2890',
        'symbol': '',
        'name': '',
        'currency': 'TWD'
    }
}
```

### operation

```
op_type (str): {
            "New":   ,
            "Cancel":   ,
            "UpdatePrice":   ,
            "UpdateQty":
        }
op_code (str): {"00":   , others:   }
op_msg (str):
```

### order

```
id (str):       trade_id
seqno (str):
ordno (str):
account (dict):
action (str):     {Buy, Sell}
price (float or int):
quantity (int):
order_type (str):      {ROD, IOC, FOK}
price_type (str): {LMT:   , MKT:   , MKP:     }
order_cond (str): {
            Cash:   ,
            MarginTrading:   ,
            ShortSelling:
        }
order_lot (str): {
            Common:   ,
            Fixing:   ,
            Odd:     ,
            IntradayOdd:
        }
custom_field (str):
```

### status

```
id (str):       trade_id
exchange_ts (int):
modified_price (float or int):
cancel_quantity (int):
order_quantity (int):
web_id (str):
```

### contract

```
security_type (str):
exchange (str):
code (str):
symbol (str):
name (str):
currency (str):
```

`id` `trade_id`

**version>=1.0** **version<1.0**

```
OrderState.StockDeal {
    'trade_id': '9c6ae2eb',
    'seqno': '269866',
    'ordno': 'IN497',
    'exchange_seq': '669915',
    'broker_id': '9A95',
    'account_id': '1234567',
    'action': 'Buy',
    'code': '2890',
    'order_cond': 'Cash',
    'order_lot': 'IntradayOdd',
    'price': 267.5,
    'quantity': 3,
    'web_id': '137',
    'custom_field': 'test',
    'ts': 1673577256.354
}
```

```
OrderState.TFTDeal {
    'trade_id': '9c6ae2eb',
    'seqno': '269866',
    'ordno': 'IN497',
    'exchange_seq': '669915',
    'broker_id': '9A95',
    'account_id': '1234567',
    'action': 'Buy',
    'code': '2890',
    'order_cond': 'Cash',
    'order_lot': 'IntradayOdd',
    'price': 267.5,
    'quantity': 3,
    'web_id': '137',
    'custom_field': 'test',
    'ts': 1673577256.354
}
```

```
trade_id (str):      id
seqno (str):
ordno (str):
exchange_seq (str):
broker_id (str):
account_id (str):
action (str):     {Buy, Sell}
code (str):
order_cond (str): {
            Cash:   ,
            MarginTrading:   ,
            ShortSelling:
        }
order_lot (str): {
            Common:   ,
            Fixing:   ,
            Odd:     ,
            IntradayOdd:
        }
price (float or int):
quantity (int):
web_id (str):
custom_field (str):
ts (int):
```

Callback

operation order status contract

**version>=1.0** **version<1.0**

```
OrderState.FuturesOrder {
    'operation': {
        'op_type': 'New',
        'op_code': '00',
        'op_msg': ''
    },
    'order': {
        'id': 'fcb42a6e',
        'seqno': '585886',
        'ordno': '00',
        'account': {
            'account_type': 'F',
            'person_id': '',
            'broker_id': 'F002000',
            'account_id': '1234567',
            'signed': True
        },
        'action': 'Buy',
        'price': 14000.0,
        'quantity': 1,
        'order_type': 'ROD',
        'price_type': 'LMT',
        'market_type': 'Night',
        'oc_type': 'New',
        'subaccount': '',
        'combo': False
    },
    'status': {
        'id': 'fcb42a6e',
        'exchange_ts': 1673512283.0,
        'modified_price': 0.0,
        'cancel_quantity': 0,
        'order_quantity': 1,
        'web_id': 'Z'
    },
    'contract': {
        'security_type': 'FUT',
        'code': 'TXF',
        'exchange': 'TIM',
        'delivery_month': '202301',
        'delivery_date': '',
        'strike_price': 0.0,
        'option_right': 'Future'
    }
}

OrderState.FOrder {
    'operation': {
        'op_type': 'New',
        'op_code': '00',
        'op_msg': ''
    },
    'order': {
        'id': 'fcb42a6e',
        'seqno': '585886',
        'ordno': '00',
        'account': {
            'account_type': 'F',
            'person_id': '',
            'broker_id': 'F002000',
            'account_id': '1234567',
            'signed': True
        },
        'action': 'Buy',
        'price': 14000.0,
        'quantity': 1,
        'order_type': 'ROD',
        'price_type': 'LMT',
        'market_type': 'Night',
        'oc_type': 'New',
        'subaccount': '',
        'combo': False
    },
    'status': {
        'id': 'fcb42a6e',
        'exchange_ts': 1673512283.0,
        'modified_price': 0.0,
        'cancel_quantity': 0,
        'order_quantity': 1,
        'web_id': 'Z'
    },
    'contract': {
        'security_type': 'FUT',
        'code': 'TXF',
        'exchange': 'TIM',
        'delivery_month': '202301',
        'delivery_date': '',
        'strike_price': 0.0,
        'option_right': 'Future'
    }
}
```

### operation

```
op_type (str): {
        "New":   ,
        "Cancel":   ,
        "UpdatePrice":   ,
        "UpdateQty":
    }
op_code (str): {"00":   , others:   }
op_msg (str):
```

### order

```
id (str):       trade_id
seqno (str):
ordno (str):
account (dict):
action (str):
price (float or int):
quantity (int):
order_cond (str): {
            Cash:   ,
            MarginTrading:   ,
            ShortSelling:
        }
order_type (str):      {ROD, IOC, FOK}
price_type (str): {LMT:   , MKT:   , MKP:     }
market_type (str):     {Day:  , Night:  }
oc_type(str): {New:   , Cover:   , Auto:   }
subaccount(str):
combo (bool):
```

### status

```
id (str):       trade_id
exchange_ts (int):
modified_price (float or int):
cancel_quantity (int):
order_quantity (int):
web_id (str):
```

### contract

```
security_type (str):
code (str):
exchange (str):
delivery_month (str):
delivery_date (str):
strike_price (float):
option_right (str): {Future, OptionCall, OptionPut}
```

`id` `trade_id`

**version>=1.0** **version<1.0**

```
OrderState.FuturesDeal {
    'trade_id': '4e6df0f6',
    'seqno': '458545',
    'ordno': 'tA0deX1O',
    'exchange_seq': 'j5006396',
    'broker_id': 'F002000',
    'account_id': '1234567',
    'action': 'Sell',
    'code': 'TXO',
    'price': 58.0,
    'quantity': 1,
    'subaccount': '',
    'security_type': 'OPT',
    'delivery_month': '202301',
    'strike_price': 14300.0,
    'option_right': 'OptionPut',
    'market_type': 'Day',
    'combo': False,
    'ts': 1673270852.0
}

OrderState.FDeal {
    'trade_id': '4e6df0f6',
    'seqno': '458545',
    'ordno': 'tA0deX1O',
    'exchange_seq': 'j5006396',
    'broker_id': 'F002000',
    'account_id': '1234567',
    'action': 'Sell',
    'code': 'TXO',
    'price': 58.0,
    'quantity': 1,
    'subaccount': '',
    'security_type': 'OPT',
    'delivery_month': '202301',
    'strike_price': 14300.0,
    'option_right': 'OptionPut',
    'market_type': 'Day',
    'combo': False,
    'ts': 1673270852.0
}
```

```
trade_id (str):      id
seqno (str):
ordno (str):
exchange_seq (str):
broker_id (str):
account_id (str):
action (str):
code (str):
price (float or int):
quantity (int):
subaccount (str):
security_type (str):
delivery_month (str):
strike_price (float):
option_right (str): {Future, OptionCall, OptionPut}
market_type (str): {Day, Night}
ts (int):
```

Callback

# 4.6 CallBack

## 4.6.1

`place_order` `update_order` `cancel_order`

`set_order_callback` `order_cb` ) print `my_order_callback` print

```
def order_cb(stat, msg):
    print('my_order_callback')
    print(stat, msg)

api.set_order_callback(order_cb)
```

**version>=1.0** **version<1.0**

```
contract = api.Contracts.Stocks.TSE.TSE2890
order = api.Order(
    price=16,
    quantity=1,
    action=sj.constant.Action.Buy,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    order_lot=sj.constant.StockOrderLot.Common,
    custom_field="test",
    account=api.stock_account
)
trade = api.place_order(contract, order)
```

```
contract = api.Contracts.Stocks.TSE.TSE2890
order = api.Order(
    price=16,
    quantity=1,
    action=sj.constant.Action.Buy,
    price_type=sj.constant.TFTStockPriceType.LMT,
    order_type=sj.constant.TFTOrderType.ROD,
    order_lot=sj.constant.TFTStockOrderLot.Common,
    custom_field="test",
    account=api.stock_account
)
trade = api.place_order(contract, order)
```

**version>=1.0** **version<1.0**

```
my_order_callback
OrderState.StockOrder {
    'operation': {
        'op_type': 'New',
        'op_code': '00',
        'op_msg': ''
    },
    'order': {
        'id': '97b63e2f',
        'seqno': '267677',
        'ordno': 'IM394',
        'account': {
            'account_type': 'S',
            'person_id': '',
            'broker_id': '9A95',
            'account_id': '1234567',
            'signed': True
        },
        'action': 'Buy',
        'price': 16.0,
        'quantity': 1,
        'order_type': 'ROD',
        'price_type': 'LMT',
        'order_cond': 'Cash',
        'order_lot': 'Common',
        'custom_field': 'test'
    },
    'status': {
        'id': '97b63e2f',
        'exchange_ts': 1673576134.038,
        'modified_price': 0.0,
        'cancel_quantity': 0,
        'order_quantity': 1,
        'web_id': '137'
    },
    'contract': {
        'security_type': 'STK',
        'exchange': 'TSE',
        'code': '2890',
        'symbol': '',
        'name': '',
        'currency': 'TWD'
    }
}
```

```
my_order_callback
OrderState.TFTOrder {
    'operation': {
        'op_type': 'New',
        'op_code': '00',
        'op_msg': ''
    },
    'order': {
        'id': '97b63e2f',
        'seqno': '267677',
        'ordno': 'IM394',
        'account': {
            'account_type': 'S',
            'person_id': '',
            'broker_id': '9A95',
            'account_id': '1234567',
            'signed': True
        },
        'action': 'Buy',
        'price': 16.0,
        'quantity': 1,
        'order_type': 'ROD',
        'price_type': 'LMT',
        'order_cond': 'Cash',
        'order_lot': 'Common',
        'custom_field': 'test'
    },
    'status': {
        'id': '97b63e2f',
        'exchange_ts': 1673576134.038,
        'modified_price': 0.0,
        'cancel_quantity': 0,
        'order_quantity': 1,
        'web_id': '137'
    },
    'contract': {
        'security_type': 'STK',
        'exchange': 'TSE',
        'code': '2890',
        'symbol': '',
        'name': '',
        'currency': 'TWD'
    }
}
```

**version>=1.0** **version<1.0**

```
my_order_callback
OrderState.StockDeal {
    'trade_id': '9c6ae2eb',
    'seqno': '269866',
    'ordno': 'IN497',
    'exchange_seq': '669915',
    'broker_id': '9A95',
    'account_id': '1234567',
    'action': 'Buy',
    'code': '2890',
    'order_cond': 'Cash',
    'order_lot': 'IntradayOdd',
    'price': 267.5,
    'quantity': 3,
    'web_id': '137',
    'custom_field': 'test',
    'ts': 1673577256.354
}
```

```
my_order_callback
OrderState.TFTDeal {
    'trade_id': '9c6ae2eb',
    'seqno': '269866',
    'ordno': 'IN497',
    'exchange_seq': '669915',
    'broker_id': '9A95',
    'account_id': '1234567',
    'action': 'Buy',
    'code': '2890',
    'order_cond': 'Cash',
    'order_lot': 'IntradayOdd',
    'price': 267.5,
    'quantity': 3,
    'web_id': '137',
    'custom_field': 'test',
    'ts': 1673577256.354
}
```

## 4.6.2

solace mesh broker solace 50

In

```
@api.quote.on_event
def event_callback(resp_code: int, event_code: int, info: str, event: str):
    print(f'Event code: {event_code} | Event: {event}')
```

Out

```
Event code: 16 | Event: Subscribe or Unsubscribe ok
```

callback callback

In

```
api.quote.set_event_callback?
```

Out

```
Signature:
    api.quote.set_event_callback(func:Callable[[int, int, str, str], NoneType]) -> None
Docstring:  <no docstring>
Type:  method
```

| Event Code | Event Code Enumerator | Description |
|---|---|---|
| 0 | SOLCLIENT_SESSION_EVENT_UP_NOTICE | The Session is established. |
| 1 | SOLCLIENT_SESSION_EVENT_DOWN_ERROR | The Session was established and then went down. |
| 2 | SOLCLIENT_SESSION_EVENT_CONNECT_FAILED_ERROR | The Session attempted to connect but was unsucces |
| 3 | SOLCLIENT_SESSION_EVENT_REJECTED_MSG_ERROR | The appliance rejected a published message. |
| 4 | SOLCLIENT_SESSION_EVENT_SUBSCRIPTION_ERROR | The appliance rejected a subscription (add or remov |
| 5 | SOLCLIENT_SESSION_EVENT_RX_MSG_TOO_BIG_ERROR | The API discarded a received message that exceede Session buffer size. |
| 6 | SOLCLIENT_SESSION_EVENT_ACKNOWLEDGEMENT | The oldest transmitted Persistent/Non-Persistent me that has been acknowledged. |
| 7 | SOLCLIENT_SESSION_EVENT_ASSURED_PUBLISHING_UP | Deprecated -- see notes in solClient_session_startAssuredPublishing.The AD Ha (that is, Guaranteed Delivery handshake) has compl the publisher and Guaranteed messages can be sent |
| 8 | SOLCLIENT_SESSION_EVENT_ASSURED_CONNECT_FAILED | Deprecated -- see notes in solClient_session_startAssuredPublishing.The applia rejected the AD Handshake to start Guaranteed pub Use SOLCLIENT_SESSION_EVENT_ASSURED_DELIVER instead. |
| 8 | SOLCLIENT_SESSION_EVENT_ASSURED_DELIVERY_DOWN | Guaranteed Delivery publishing is not available.The guaranteed delivery capability on the session has be disabled by some action on the appliance. |
| 9 | SOLCLIENT_SESSION_EVENT_TE_UNSUBSCRIBE_ERROR | The Topic Endpoint unsubscribe command failed. |
| 9 | SOLCLIENT_SESSION_EVENT_DTE_UNSUBSCRIBE_ERROR | Deprecated name; SOLCLIENT_SESSION_EVENT_TE_UNSUBSCRIBE_ is preferred. |
| 10 | SOLCLIENT_SESSION_EVENT_TE_UNSUBSCRIBE_OK | The Topic Endpoint unsubscribe completed. |
| 10 | SOLCLIENT_SESSION_EVENT_DTE_UNSUBSCRIBE_OK | Deprecated name; SOLCLIENT_SESSION_EVENT_TE_UNSUBSCRIBE_ preferred. |
| 11 | SOLCLIENT_SESSION_EVENT_CAN_SEND | The send is no longer blocked. |
| 12 | SOLCLIENT_SESSION_EVENT_RECONNECTING_NOTICE | The Session has gone down, and an automatic recon attempt is in progress. |
| 13 | SOLCLIENT_SESSION_EVENT_RECONNECTED_NOTICE | The automatic reconnect of the Session was success the Session was established again. |
| 14 | SOLCLIENT_SESSION_EVENT_PROVISION_ERROR | The endpoint create/delete command failed. |
| 15 | SOLCLIENT_SESSION_EVENT_PROVISION_OK | The endpoint create/delete command completed. |
| 16 | SOLCLIENT_SESSION_EVENT_SUBSCRIPTION_OK | The subscribe or unsubscribe operation has succeed |
| 17 | SOLCLIENT_SESSION_EVENT_VIRTUAL_ROUTER_NAME_CHANGED | The appliance's Virtual Router Name changed durin reconnect operation.This could render existing queu temporary topics invalid. |
| 18 | SOLCLIENT_SESSION_EVENT_MODIFYPROP_OK | The session property modification completed. |
| 19 | SOLCLIENT_SESSION_EVENT_MODIFYPROP_FAIL | The session property modification failed. |

| Event Code | Event Code Enumerator | Description |
| --- | --- | --- |
| 20 | SOLCLIENT_SESSION_EVENT_REPUBLISH_UNACKED_MESSAGES | After successfully reconnecting a disconnected sess SDK received an unknown publisher flow name resp when reconnecting the GD publisher flow. |

# 4.7

## 4.7.1

**In**

```
api.account_balance?
```

**Out**

```
Signature:
    api.account_balance(
        timeout: int = 5000,
        cb: Callable[[shioaji.position.AccountBalance], NoneType] = None,
    )
Docstring: query stock account balance
```

**In**

```
api.account_balance()
```

**Out**

```
AccountBalance(
    status=<FetchStatus.Fetched: 'Fetched'>,
    acc_balance=100000.0,
    date='2023-01-06 13:30:00.000000',
    errmsg=''
)
```

**AccountBalance**

```
status (FetchStatus):
acc_balance (float):
date (str):
errmsg (str):
```

## 4.7.2

In

```
api.margin?
```

Out

```
Signature:
api.margin(
    account: shioaji.account.Account = None,
    timeout: int = 5000,
    cb: Callable[[shioaji.position.Margin], NoneType] = None,
) -> shioaji.position.Margin
Docstring: query future account of margin
```

In

```
api.margin(api.futopt_account)
```

Out

```
Margin(
    status=<FetchStatus.Fetched: 'Fetched'>,
    yesterday_balance=6000.0,
    today_balance=6000.0,
    deposit_withdrawal=0.0,
    fee=0.0,
    tax=0.0,
    initial_margin=0.0,
    maintenance_margin=0.0,
    margin_call=0.0,
    risk_indicator=999.0,
    royalty_revenue_expenditure=0.0,
    equity=6000.0,
    equity_amount=6000.0,
    option_openbuy_market_value=0.0,
    option_opensell_market_value=0.0,
    option_open_position=0.0,
    option_settle_profitloss=0.0,
    future_open_position=0.0,
    today_future_open_position=0.0,
    future_settle_profitloss=0.0,
    available_margin=6000.0,
    plus_margin=0.0,
    plus_margin_indicator=0.0,
    security_collateral_amount=0.0,
    order_margin_premium=0.0,
    collateral_amount=0.0
)
```

### Margin

```
status (FetchStatus):
yesterday_balance (float):
today_balance (float):
deposit_withdrawal (float):
fee (float):
tax (float):
initial_margin (float):
maintenance_margin (float):
margin_call (float):
risk_indicator (float):
royalty_revenue_expenditure (float):
equity (float):
equity_amount (float):
option_openbuy_market_value (float):
option_opensell_market_value (float):
option_open_position (float):
option_settle_profitloss (float):
future_open_position (float):
today_future_open_position (float):
future_settle_profitloss (float):
available_margin (float):    (  )
plus_margin (float):
plus_margin_indicator (float):
security_collateral_amount (float):
order_margin_premium (float):
collateral_amount (float):
```

## 4.7.3

In

```
api.list_positions?
```

Out

```
Signature:
api.list_positions(
    account: shioaji.account.Account = None,
    unit: shioaji.constant.Unit = <Unit.Common: 'Common'>,
    timeout: int = 5000,
    cb: Callable[[List[Union[shioaji.position.StockPosition, shioaji.position.FuturePosition]]], NoneType] = None,
) -> List[Union[shioaji.position.StockPosition, shioaji.position.FuturePosition]]
Docstring:
query account of unrealized gain or loss
Args:
    account (:obj:Account):
        choice the account from listing account (Default: stock account)
```

In

```
api.list_positions(api.stock_account)
```

Out

```
[
    StockPosition(
        id=0,
        code='2890',
        direction=<Action.Buy: 'Buy'>,
        quantity=12,
        price=2.79,
        last_price=16.95,
        pnl=169171.0,
        yd_quantity=12,
        margin_purchase_amount=0,
        collateral=0,
        short_sale_margin=0,
        interest=0
    )
]
```

DataFrame

In

```
positions = api.list_positions(api.stock_account)
df = pd.DataFrame(s.__dict__ for s in positions)
df
```

**Out**

| id | code | direction | quantity | price | last_price | pnl | yd_quantity | cond | margi |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2890 | Buy | 12 | 2.79 | 16.95 | 169172 | 12 | |

**StockPosition**

```
id (int):
code (str):
direction (Action): {Buy:  , Sell:  }
quantity (int):
price (float):
last_price (float):
pnl (float):
yd_quantity (int):
cond (StockOrderCond): {
    Cash:   (   ),
    Netting:     ,
    MarginTrading:   ,
    ShortSelling:   ,
    Emerging:
    }
margin_purchase_amount (int):
collateral (int):
short_sale_margin (int):
interest (int):
```

**In**

```
api.list_positions(
    api.stock_account,
    unit=sj.constant.Unit.Share
)
```

**Out**

```
[
    StockPosition(
        id=0,
        code='2890',
        direction=<Action.Buy: 'Buy'>,
        quantity=10000,
        price=10.1,
        last_price=12.0,
        pnl=1234.0,
        yd_quantity=10000,
        margin_purchase_amount=0,
        collateral=0,
        short_sale_margin=0,
        interest=0
    )
]
```

`account`

**In**

```
api.list_positions(api.futopt_account)
```

**Out**

```
[
    FuturePosition(
        id=0,
        code='TX201370J2',
        direction=<Action.Buy: 'Buy'>,
        quantity=3,
        price=131.0000,
        last_price=126.0,
        pnl=-750.00
    )
]
```

DataFrame

**In**

```
positions = api.list_positions(api.futopt_account)
df = pd.DataFrame(p.__dict__ for p in positions)
df
```

**Out**

| | id | code | direction | quantity | price | last_price | pnl |
|---|---|---|---|---|---|---|---|
| 0 | 0 | TXFA3 | Buy | 4 | 14181 | 14375 | 155200 |

**FuturePosition**

```
id (int):
code (str):
direction (Action): {Buy:  , Sell:  }
quantity (int):
price (float):
last_price (float):
pnl (float):
```

-

`list_positions` `id` `detail_id`

**In**

```
api.list_position_detail?
```

**Out**

```
Signature:
    api.list_position_detail(
    account: shioaji.account.Account = None,
    detail_id: int = 0,
    timeout: int = 5000,
    cb: Callable[[List[Union[shioaji.position.StockPositionDetail, shioaji.position.    FuturePositionDetail]]], NoneType] = None,
) -> List[Union[shioaji.position.StockPositionDetail, shioaji.position.FuturePositionDetail]]
Docstring:
query account of position detail

Args:
    account (:obj:Account):
        choice the account from listing account (Default: stock account)
    detail_id (int): the id is from Position object, Position is from list_positions
```

In

```
position_detail = api.list_position_detail(api.stock_account, 1)
position_detail
```

Out

```
[
    StockPositionDetail(
        date='2023-02-22',
        code='3558',
        quantity=0,
        price=1461.0,
        last_price=1470.0,
        dseq='WA371',
        direction=<Action.Buy: 'Buy'>,
        pnl=9.0,
        currency=<Currency.TWD: 'TWD'>,
        fee=1.0
    )
]
```

DataFrame

In

```
df = pd.DataFrame(pnl.__dict__ for pnl in position_detail)
df
```

Out

| date | code | quantity | price | last_price | direction | pnl | currency | fee |
|---|---|---|---|---|---|---|---|---|
| 2023-02-22 | 3558 | 0 | 1461.0 | WA371 | Action.Buy | 11.0 | Currency.TWD | 1.0 |

```
date (str):
code (str):
quantity (int):
price (float):
last_price (float):
dseq (str):
direction (Action): {Buy:  , Sell:  }
pnl (decimal):
currency (string):    {NTD, USD, HKD, EUR, CAD, BAS}
fee (decimal):
cond (StockOrderCond): {
    Cash:   (   ),
    Netting:     ,
    MarginTrading:   ,
    ShortSelling:   ,
    Emerging:
    }
ex_dividends(int):
interest (int):
margintrading_amt(int):
collateral (int):
```

In

```
position_detail = api.list_position_detail(api.futopt_account, 0)
position_detail
```

**Out**

```
[
    FuturePositionDetail(
        date='2023-02-14',
        code='MXFC3',
        quantity=1,
        price=15611.0,
        last_price=15541.0,
        dseq='tA0n8',
        direction=<Action.Buy: 'Buy'>,
        pnl=-3500.0,
        currency=<Currency.TWD: 'TWD'>,
        entry_quantity=1
    )
]
```

DataFrame

**In**

```
df = pd.DataFrame(pnl.__dict__ for pnl in position_detail)
df
```

**Out**

| date | code | quantity | price | last_price | dseq | direction | pnl | curr |
|---|---|---|---|---|---|---|---|---|
| 2023-02-14 | MXFC3 | 1 | 15611.0 | 15541.0 | tA0n8 | Action.Buy | -3500.0 | Currency. |

```
code (str):
date (str):
quantity (int):
price (float):
last_price (float):
dseq (str):
direction (Action): {Buy:  , Sell:  }
pnl (float):
currency (str):    {NTD, USD, HKD, EUR, CAD, BAS}
fee (float or int):
entry_quantity(int):
```

## 4.7.4

In

```
api.list_profit_loss?
```

Out

```
Signature:
    api.list_profit_loss(
        account: shioaji.account.Account = None,
        begin_date: str = '',
        end_date: str = '',
        unit: shioaji.constant.Unit = <Unit.Common: 'Common'>,
        timeout: int = 5000,
        cb: Callable[[List[shioaji.position.ProfitLoss]], NoneType] = None,
    ) -> List[shioaji.position.ProfitLoss]
Docstring:
    query account of profit loss

    Args:
        account (:obj:Account):
            choice the account from listing account (Default: stock account)
        begin_date (str): the start date of query profit loss (Default: today)
        end_date (str): the end date of query profit loss (Default: today)
```

`begin_date` `end_date` `unit` `Common` `Share`

In

```
profitloss = api.list_profit_loss(api.stock_account,'2020-05-05','2020-05-30')
profitloss
```

Out

```
[
    StockProfitLoss(
        id=0,
        code='2890',
        seqno='14816',
        dseq='ID111',
        quantity=1,
        price=10.1,
        pnl=1234.0,
        pr_ratio=0.1237,
        cond='Cash',
        date='2020-05-22'
    )
]
```

DataFrame

In

```
df = pd.DataFrame(pnl.__dict__ for pnl in profitloss)
df
```

**Out**

| id | code | cond | date | pnl | pr_ratio | price | quantity | seqno |
|---|---|---|---|---|---|---|---|---|
| 0 | 2890 | StockOrderCond.Cash | 2020-05-22 | 1000.0 | 0.1237 | 10.1 | 1 | 14816 |

**StockProfitLoss**

```
id (int):     id
code (str):
seqno (str): seqno no.
dseq (str): seqno no.
quantity (int):
price (float):
pnl (float):
pr_ratio (float):
cond (StockOrderCond): {
                Cash:   (   ),
                Netting:     ,
                MarginTrading:   ,
                ShortSelling:   ,
                Emerging:
            }
date (str):
```

**FutureProfitLoss**

```
id (int):     id
code (str):
quantity (int):
pnl (float):
date (str):
entry_price (int):
cover_price (int):
tax (int):
fee (int):
```

-

`list_profit_loss` `id` `detail_id` `unit` `Common` `Share`

**In**

```
api.list_profit_loss_detail?
```

**Out**

```
Signature:
api.list_profit_loss_detail(
    account: shioaji.account.Account = None,
    detail_id: int = 0,
    unit: shioaji.constant.Unit = <Unit.Common: 'Common'>,
    timeout: int = 5000,
    cb: Callable[[List[Union[shioaji.position.StockProfitDetail, shioaji.position.FutureProfitDetail]]], NoneType] = None,
) -> List[Union[shioaji.position.StockProfitDetail, shioaji.position.FutureProfitDetail]]
Docstring:
query account of profit loss detail

Args:
    account (:obj:Account):
        choice the account from listing account (Default: stock account)
    detail_id (int): the id is from ProfitLoss object, ProfitLoss is from list_profit_loss
```

**In**

```
profitloss_detail = api.list_profit_loss_detail(api.stock_account, 2)
profitloss_detail
```

**Out**

```
[
    StockProfitDetail(
        date='2020-01-01',
        code='2890',
        quantity=1,
        dseq='IX000',
        fee=20,
        tax=0,
        currency='TWD',
        price=10.8,
        cost=10820,
        rep_margintrading_amt=0,
        rep_collateral=0,
        rep_margin=0,
        shortselling_fee=0,
        ex_dividend_amt=0,
        interest=0
    )
]
```

DataFrame

**In**

```
df = pd.DataFrame(pnl.__dict__ for pnl in profitloss_detail)
df
```

**Out**

| date | code | quantity | dseq | fee | tax | currency | price | cost | rep_margint |
|---|---|---|---|---|---|---|---|---|---|
| 2020-01-01 | 2890 | 1 | IX000 | 20 | 0 | TWD | 10.8 | 10820 | |

**StockProfitDetail**

```
date (str):
code (str):
quantity (int):
dseq (str):
fee (int):
tax (int):
currency (str):    {NTD, USD, HKD, EUR, CAD, BAS}
price (float):
cost (int):
rep_margintrading_amt (int):
rep_collateral (int):
rep_margin (int):
shortselling_fee (int):
ex_dividend_amt:
interest (int):
trade_type (TradeType): {Common, DayTrade}
cond (StockOrderCond): {
                Cash:   (   ),
                Netting:     ,
                MarginTrading:   ,
                ShortSelling:   ,
                Emerging:
            }
```

**FutureProfitDetail**

```
date (str):
code (str):
quantity (int):
dseq (str):
fee (int):
tax (int):
currency (str):    {NTD, USD, HKD, EUR, CAD, BAS}
direction (Action):     {Buy, Sell}
entry_price (int):
cover_price (int):
pnl (int):
```

-

In

```
api.list_profit_loss_summary?
```

Out

```
Signature:
api.list_profit_loss_summary(
    account: shioaji.account.Account = None,
    begin_date: str = '',
    end_date: str = '',
    timeout: int = 5000,
    cb: Callable[[ProfitLossSummaryTotal], NoneType] = None,
) -> ProfitLossSummaryTotal
Docstring:
query summary profit loss of a period time

Args:
    account (:obj:Account):
        choice the account from listing account (Default: stock account)
    begin_date (str): the start date of query profit loss (Default: today)
    end_date (str): the end date of query profit loss (Default: today)
```

`begin_date` `end_date`

In

```
profitloss_summary = api.list_profit_loss_summary(api.stock_account,'2020-05-05','2020-05-30')
profitloss_summary
```

Out

```
ProfitLossSummaryTotal(
    status=<FetchStatus.Fetched: 'Fetched'>,
    profitloss_summary=[
        StockProfitLossSummary(
            code='2890',
            quantity=2000,
            entry_price=17,
            cover_price=10,
            pnl=-11585.0,
            currency='NTD',
            entry_cost=34550,
            cover_cost=21600,
            buy_cost=33112,
            sell_cost=21527,
            pr_ratio=-34.99
        )
    ],
    total=ProfitLossTotal(
        quantity=2000,
        buy_cost=33112,
        sell_cost=21527,
        pnl=-11585.0,
        pr_ratio=-34.99
    )
)
```

DataFrame

In

```
df = pd.DataFrame(pnl.__dict__ for pnl in profitloss_summary.profitloss_summary)
df
```

**Out**

| code | quantity | entry_price | cover_price | pnl | currency | entry_cost | cover_cost | bu |
|---|---|---|---|---|---|---|---|---|
| 2890 | 2000 | 17 | 10 | -11585 | NTD | 34550 | 21600 | |

**StockProfitLossSummary**

```
code (str):
quantity (int):
entry_price (int):
cover_price (int):
pnl (float):
currency (str):
entry_cost (int):     (          )
cover_cost (int):     (          )
buy_cost (int):
sell_cost (int):
pr_ratio (float):
cond (StockOrderCond): {
                Cash:   (   ),
                Netting:     ,
                MarginTrading:   ,
                ShortSelling:   ,
                Emerging:
            }
```

**FutureProfitLossSummary**

```
code (str):
quantity (int):
entry_price (int):
cover_price (int):
pnl (float):
currency (str):
direction (Action):     {Buy, Sell}
tax (int):
fee (int):
```

## 4.7.5

### Settlements

**In**

```
api.settlements?
```

**Out**

```
Signature:
api.settlements(
    account: shioaji.account.Account = None,
    timeout: int = 5000,
    cb: Callable[[List[shioaji.position.SettlementV1]], NoneType] = None,
) -> List[shioaji.position.SettlementV1]
Docstring: query stock account of settlements
```

**In**

```
settlements = api.settlements(api.stock_account)
settlements
```

**Out**

```
[
    SettlementV1(date=datetime.date(2022, 10, 13), amount=0.0, T=0),
    SettlementV1(date=datetime.date(2022, 10, 14), amount=0.0, T=1),
    SettlementV1(date=datetime.date(2022, 10, 17), amount=0.0, T=2)
]
```

DataFrame

**In**

```
df = pd.DataFrame([s.__dict__ for s in settlements]).set_index("T")
df
```

**Out**

| T | date | amount |
|---|---|---|
| 0 | 2022-10-13 | 0 |
| 1 | 2022-10-14 | 0 |
| 2 | 2022-10-17 | 0 |

**SettlementV1**

```
date (datetime.date):
amount (float):
T (int): Tday
```

# 4.8

```
import shioaji as sj
api = sj.Shioaji(simulation=True)
```

## 4.8.1 APIs

1. quote.subscribe
2. quote.unsubscribe
3. ticks
4. kbars
5. snapshots
6. short_stock_sources
7. credit_enquires
8. scanners

1. place_order
2. update_order
3. cancel_order
4. update_status
5. list_trades

1. list_positions
2. list_profit_loss

# 4.9

## 4.9.1

Shioaji Redis Stream

**In: pythonic way by using decorator**

```
from collections import defaultdict, deque
from shioaji import TickFOPv1, Exchange

# set context
msg_queue = defaultdict(deque)
api.set_context(msg_queue)

# In order to use context, set bind=True
@api.on_tick_fop_v1(bind=True)
def quote_callback(self, exchange:Exchange, tick:TickFOPv1):
    # append quote to message queue
    self[tick.code].append(tick)

# subscribe
api.quote.subscribe(
    api.Contracts.Futures.TXF['TXF202107'],
    quote_type = sj.constant.QuoteType.Tick,
    version = sj.constant.QuoteVersion.v1
)
```

**In: traditional way**

```
def quote_callback(self, exchange:Exchange, tick:TickFOPv1):
    # append tick to context
    self[tick.code].append(tick)

# In order to use context, set bind=True
api.quote.set_on_tick_fop_v1_callback(quote_callback, bind=True)
```

**Out**

```
# after subscribe and wait for a few seconds ...
# print(msg_queue)
defaultdict(collections.deque,
    {
        'TXFG1': [
            Tick(code='TXFG1', datetime=datetime.datetime(2021, 7, 5, 10, 0, 21, 220000), open=Decimal('17755'), underlying_price=Decimal('17851.88'),
bid_side_total_vol=34824, ask_side_total_vol=36212, avg_price=Decimal('17837.053112'), close=Decimal('17833'), high=Decimal('17900'), low=Decimal('17742'),
amount=Decimal('17833'), total_amount=Decimal('981323314'), volume=1, total_volume=55016, tick_type=1, chg_type=2, price_chg=Decimal('184'),
pct_chg=Decimal('1.042552'), simtrade=0),
            Tick(code='TXFG1', datetime=datetime.datetime(2021, 7, 5, 10, 0, 21, 781000), open=Decimal('17755'), underlying_price=Decimal('17851.88'),
bid_side_total_vol=34825, ask_side_total_vol=36213, avg_price=Decimal('17837.053056'), close=Decimal('17834'), high=Decimal('17900'), low=Decimal('17742'),
amount=Decimal('17834'), total_amount=Decimal('981341148'), volume=1, total_volume=55017, tick_type=1, chg_type=2, price_chg=Decimal('185'),
pct_chg=Decimal('1.048218'), simtrade=0)
        ]
    }
)
```

**REDIS STREAM**

redis

In

```
import redis
import json
from shioaji import TickFOPv1, Exchange

# redis setting
r = redis.Redis(host='localhost', port=6379, db=0, decode_responses=True)

# set up context
api.set_context(r)

# In order to use context, set bind=True
@api.on_tick_fop_v1(bind=True)
def quote_callback(self, exchange:Exchange, tick:TickFOPv1):
    # push them to redis stream
    channel = 'Q:' + tick.code # ='Q:TXFG1' in this example
    self.xadd(channel, {'tick':json.dumps(tick.to_dict(raw=True))})
```

Out

```
# after subscribe and wait for a few seconds ...
# r.xread({'Q:TXFG1':'0-0'})
[
    ['Q:TXFG1',
        [
            ('1625454940107-0',
                {'tick':
                    '{"code": "TXFG1", "datetime": "2021-07-05T11:15:49.066000", "open": "17755", "underlying_price": "17904.03", "bid_side_total_vol": 49698,
"ask_side_total_vol": 51490, "avg_price": "17851.312322", "close": "17889", "high": "17918", "low": "17742", "amount": "268335", "total_amount": "1399310819",
"volume": 15, "total_volume": 78387, "tick_type": 2, "chg_type": 2, "price_chg": "240", "pct_chg": "1.35985", "simtrade": 0}'
                }
            ),
            ('1625454941854-0',
                {'tick':
                    '{"code": "TXFG1", "datetime": "2021-07-05T11:15:50.815000", "open": "17755", "underlying_price": "17902.58", "bid_side_total_vol": 49702,
"ask_side_total_vol": 51478, "avg_price": "17851.313258", "close": "17888", "high": "17918", "low": "17742", "amount": "35776", "total_amount": "1399346595",
"volume": 2, "total_volume": 78389, "tick_type": 2, "chg_type": 2, "price_chg": "239", "pct_chg": "1.354184", "simtrade": 0}'
                }
            )
        ]
    ]
]


# parse redis stream
# [json.loads(x[-1]['tick']) for x in r.xread({'Q:TXFG1':'0-0'})[0][-1]]
[
    {
        'code': 'TXFG1',
        'datetime': '2021-07-05T11:15:49.066000',
        'open': '17755',
        'underlying_price': '17904.03',
        'bid_side_total_vol': 49698,
        'ask_side_total_vol': 51490,
        'avg_price': '17851.312322',
        'close': '17889',
        'high': '17918',
        'low': '17742',
        'amount': '268335',
        'total_amount': '1399310819',
        'volume': 15,
        'total_volume': 78387,
        'tick_type': 2,
        'chg_type': 2,
        'price_chg': '240',
        'pct_chg': '1.35985',
        'simtrade': 0
    },
    {
        'code': 'TXFG1',
        'datetime': '2021-07-05T11:15:50.815000',
        'open': '17755',
        'underlying_price': '17902.58',
        'bid_side_total_vol': 49702,
        'ask_side_total_vol': 51478,
        'avg_price': '17851.313258',
        'close': '17888',
        'high': '17918',
        'low': '17742',
        'amount': '35776',
        'total_amount': '1399346595',
        'volume': 2,
        'total_volume': 78389,
        'tick_type': 2,
        'chg_type': 2,
        'price_chg': '239',
        'pct_chg': '1.354184',
        'simtrade': 0
    },
]
```

**Example: stop order**

```
import time
from typing import Union

import shioaji as sj

class StopOrderExcecutor:
    def __init__(self, api: sj.Shioaji) -> None:
        self.api = api
        self._stop_orders = {}

    def on_quote(
        self, quote: Union[sj.BidAskFOPv1, sj.BidAskSTKv1, sj.TickFOPv1, sj.TickSTKv1]
    ) -> None:
        code = quote.code
        if code in self._stop_orders:
            for stop_order in self._stop_orders[code]:
                if stop_order['executed']:
                    continue
                if hasattr(quote, "ask_price"):
                    price = 0.5 * float(
                        quote.bid_price[0] + quote.ask_price[0]
                    )  # mid price
                else:
                    price = float(quote.close)  # Tick

                is_execute = False
                if stop_order["stop_price"] >= stop_order["ref_price"]:
                    if price >= stop_order["stop_price"]:
                        is_execute = True

                elif stop_order["stop_price"] < stop_order["ref_price"]:
                    if price <= stop_order["stop_price"]:
                        is_execute = True

                if is_execute:
                    self.api.place_order(stop_order["contract"], stop_order["pending_order"])
                    stop_order['executed'] = True
                    stop_order['ts_executed'] = time.time()
                    print(f"execute stop order: {stop_order}")
                else:
                    self._stop_orders[code]

    def add_stop_order(
        self,
        contract: sj.contracts.Contract,
        stop_price: float,
        order: sj.order.Order,
    ) -> None:
        code = contract.code
        snap = self.api.snapshots([contract])[0]
        # use mid price as current price to avoid illiquidity
        ref_price = 0.5 * (snap.buy_price + snap.sell_price)
        stop_order = {
            "code": contract.code,
            "stop_price": stop_price,
            "ref_price": ref_price,
            "contract": contract,
            "pending_order": order,
            "ts_create": time.time(),
            "executed": False,
            "ts_executed": 0.0
        }

        if code not in self._stop_orders:
            self._stop_orders[code] = []
        self._stop_orders[code].append(stop_order)
        print(f"add stop order: {stop_order}")

    def get_stop_orders(self) -> dict:
        return self._stop_orders

    def cancel_stop_order_by_code(self, code: str) -> None:
        if code in self._stop_orders:
            _ = self._stop_orders.pop(code)

    def cancel_stop_order(self, stop_order: dict) -> None:
        code = stop_order["code"]
        if code in self._stop_orders:
            self._stop_orders[code].remove(stop_order)
            if len(self._stop_orders[code]) == 0:
                self._stop_orders.pop(code)

    def cancel_all_stop_orders(self) -> None:
        self._stop_orders.clear()
```

- snapshots

(Stop-Limit Order)

**Set up a stop order**

```
# shioaji order
contract = api.Contracts.Futures.TXF['TXF202301']
order = api.Order(
    action='Buy',
    price=14800,
    quantity=1,
    price_type='LMT',
    order_type='ROD',
    octype=sj.constant.FuturesOCType.Auto,
    account=api.futopt_account
)

# Stop Order Excecutor
soe = StopOrderExcecutor(api)
soe.add_stop_order(contract=contract, stop_price=14805, order=order)
```

**Out**

```
add stop order: {
    'code': 'TXFA3',
    'stop_price': 14805,
    'ref_price': 14790,
    'contract': Future(
        code='TXFA3',
        symbol='TXF202301',
        name='    01',
        category='TXF',
        delivery_month='202301',
        delivery_date='2023/01/30',
        underlying_kind='I',
        unit=1,
        limit_up=16241.0,
        limit_down=13289.0,
        reference=14765.0,
        update_date='2023/01/10'
    ),
    'pending_order': Order(
        action=<Action.Buy: 'Buy'>,
        price=14800,
        quantity=1,
        account=FutureAccount(person_id='A123456789', broker_id='F002000', account_id='1234567', signed=True, username='PAIUSER'),
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>
    ),
    'ts_create': 1673329115.1056178,
    'executed': False,
    'ts_executed': 0.0
}
```

- (Stop-Market Order): `price_type = 'MKT'`

`StopOrderExcecutor`

**Set up context and callback function**

```
from shioaji import TickFOPv1, Exchange

# set up context
api.set_context(soe)

# In order to use context, set bind=True
@api.on_tick_fop_v1(bind=True)
def quote_callback(self, exchange:Exchange, tick:TickFOPv1):
    # pass tick object to Stop Order Excecutor
    self.on_quote(tick)

# subscribe
api.quote.subscribe(
    contract,
    quote_type = sj.constant.QuoteType.Tick,
    version = sj.constant.QuoteVersion.v1
)
```

**Out: Once close/mid price hit stop price**

```
execute stop order: {
    'code': 'TXFA3',
    'stop_price': 14805,
    'ref_price': 14790,
    'contract': Future(
        code='TXFA3',
        symbol='TXF202301',
        name='    01',
        category='TXF',
        delivery_month='202301',
        delivery_date='2023/01/30',
        underlying_kind='I',
        unit=1,
        limit_up=16241.0,
        limit_down=13289.0,
        reference=14765.0,
        update_date='2023/01/10'
    ),
    'pending_order': Order(
        action=<Action.Buy: 'Buy'>,
        price=14800,
        quantity=1,
        account=FutureAccount(person_id='A123456789', broker_id='F002000', account_id='1234567', signed=True, username='PAIUSER'),
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>
    ),
    'ts_create': 1673329115.1056178,
    'executed': True,
    'ts_executed': 1673329161.3224185
}
```

## 4.9.2

(Blocking) I/O CPU
(non-blocking)

/ `timeout` API `timeout` `0` `timeout` 5000 5

`place_order` `timeout = 0`

In

```
contract = api.Contracts.Futures.TXF['TXF202301']
order = api.Order(
    action=sj.constant.Action.Sell,
    price=14000,
    quantity=1,
    price_type=sj.constant.FuturesPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    octype=sj.constant.FuturesOCType.Auto,
    account=api.futopt_account
)
trade = api.place_order(contract, order, timeout=0)
trade
```

Out

```
Trade(
    contract=Future(
        code='TXFA3',
        symbol='TXF202301',
        name='    01',
        category='TXF',
        delivery_month='202301',
        delivery_date='2023/01/30',
        underlying_kind='I',
        unit=1,
        limit_up=16241.0,
        limit_down=13289.0,
        reference=14765.0,
        update_date='2023/01/10'
    ),
    order=Order(
        action=<Action.Sell: 'Sell'>,
        price=14000,
        quantity=1,
        account=FutureAccount(
            person_id='F123456789',
            broker_id='F002000',
            account_id='1234567',
            signed=True,
            username='PAPIUSER'
        ),
        price_type=<StockPriceType.LMT: 'LMT'>,
        order_type=<OrderType.ROD: 'ROD'>
    ),
    status=OrderStatus(status=<Status.Inactive: 'Inactive'>)
)
```

`Trade` `Order` `id` `seqno` `OrderStatus` `id` `status_code` `order_datetime`
`deals` `status` `Inactive`

**Out**

```
OrderState.FuturesOrder {
    'operation': {
        'op_type': 'New',
        'op_code': '00',
        'op_msg': ''
    },
    'order': {
        'id': 'de616839',
        'seqno': '500009',
        'ordno': '000009',
        'action': 'Sell',
        'price': 14000,
        'quantity': 1,
        'order_type': 'ROD',
        'price_type': 'LMT',
        'oc_type': 'Auto',
        'custom_field': ''
    },
    'status': {
        'id': 'de616839',
        'exchange_ts': 1673334371.492948,
        'order_quantity': 1,
        'modified_price': 0,
        'cancel_quantity': 0,
        'web_id': 'Z'
    },
    'contract': {
        'security_type': 'FUT',
        'exchange': 'TAIFEX',
        'code': 'TXFA3'
    }
}
```

**In**

```
from shioaji.order import Trade

def non_blocking_cb(trade:Trade):
    print('__my_callback__')
    print(trade)

trade = api.place_order(
    contract,
    order,
    timeout=0,
    cb=non_blocking_cb # only work in non-blocking mode
)
```

**Out: place order callback**

```
__my_callback__
contract=Future(
    code='TXFA3',
    symbol='TXF202301',
    name='    01',
    category='TXF',
    delivery_month='202301',
    delivery_date='2023/01/30',
    underlying_kind='I',
    unit=1,
    limit_up=16241.0,
    limit_down=13289.0,
    reference=14765.0,
    update_date='2023/01/10'
),
order=Order(
    action=<Action.Sell: 'Sell'>,
    price=14000,
    quantity=1,
    id='40fd85d6',
    seqno='958433',
    ordno='kY01g',
    account=FutureAccount(
        person_id='F123456789',
        broker_id='F002000',
        account_id='1234567',
        signed=True,
        username='PAPIUSER'
    ),
    price_type=<StockPriceType.LMT: 'LMT'>,
    order_type=<OrderType.ROD: 'ROD'>
),
status=OrderStatus(
    id='40fd85d6',
    status=<Status.Submitted: 'Submitted'>,
    status_code='    ',
    order_datetime=datetime.datetime(2023, 01, 10, 15, 14, 32),
    deals=[]
)
```

`place_order` 0.01 12

**contract and order**

```
contract = api.Contracts.Futures.TXF['TXF202301']
order = api.Order(
    action='Sell',
    price=14000,
    quantity=1,
    price_type='LMT',
    order_type='ROD',
    octype=sj.constant.FuturesOCType.Auto,
    account=api.futopt_account
)
```

**Blocking**

```
start_time = time.time()
api.place_order(contract, order) # block and wait for the order response
print(time.time() - start_time)
# 0.136578369140625 <- may be different
```

**Non-Blocking**

```
start_time = time.time()
api.place_order(contract, order, timeout=0) # non-block, the order is in transmition (inactive).
print(time.time() - start_time)
# 0.011670351028442383 <- may be different
```

```
- place_order
- update_order
- cancel_order
- update_status
- list_positions
- list_position_detail
- list_profit_loss
- list_profit_loss_detail
- list_profit_loss_summary
- settlements
- margin
- ticks
- kbars
```

## 4.9.3

```
from pydantic import BaseModel

class TouchOrderCond(BaseModel):
    contract: Contract
    order: Order
    order: Order
    touch_price: float

class TouchOrder:
   def __init__(self, api: sj.Shioaji, condition: TouchOrderCond
    ):
       self.flag = False
       self.api = api
       self.order = condition.order
       self.contract = condition.contract
       self.touch_price = condition.touch_price
       self.api.quote.subscribe(self.contract)
       self.api.quote.set_quote_callback(self.touch)

   def touch(self, topic, quote):
       price = quote["Close"][0]
       if price == self.touch_price and not self.flag:
           self.flag = True
           self.api.place_order(self.contract, self.order)
           self.api.quote.unsubscribe(self.contract)
```

TouchPrice Order Extention

## 4.9.4

sj-trading jupyter notebook quote_manager_usage

uv uv uv

Polars Polars polars_talib

**Polars**

```
uv add polars polars_talib
```

Polars Polars

polars_talib Polars polars expression ta-lib Polars shioaji polars_ta_extension

Polars DataFrame Shioaji Polars ticks K

**quote.py**

src/sj_trading/ quote.py

```
import shioaji as sj
from typing import List

class QuoteManager:
    def __init__(self, api: sj.Shioaji):
        self.api = api
        self.api.quote.set_on_tick_stk_v1_callback(self.on_stk_v1_tick_handler)
        self.api.quote.set_on_tick_fop_v1_callback(self.on_fop_v1_tick_handler)
        self.ticks_stk_v1: List[sj.TickSTKv1] = []
        self.ticks_fop_v1: List[sj.TickFOPv1] = []

    def on_stk_v1_tick_handler(self, _exchange: sj.Exchange, tick: sj.TickSTKv1):
        self.ticks_stk_v1.append(tick)

    def on_fop_v1_tick_handler(self, _exchange: sj.Exchange, tick: sj.TickFOPv1):
        self.ticks_fop_v1.append(tick)
```

handle func QuoteManager on_stk_v1_tick_handler on_fop_v1_tick_handler ticks_stk_v1 ticks_fop_v1

**QuoteManager**

```
def __init__(self, api: sj.Shioaji):
    # skip
    self.subscribed_stk_tick: Set[str] = set()

def subscribe_stk_tick(self, codes: List[str], recover: bool = False):
    for code in codes:
        contract = self.api.Contracts.Stocks[code]
        if contract is not None and code not in self.subscribed_stk_tick:
            self.api.quote.subscribe(contract, "tick")
            self.subscribed_stk_tick.add(code)

def unsubscribe_stk_tick(self, codes: List[str]):
    for code in codes:
        contract = self.api.Contracts.Stocks[code]
        if contract is not None and code in self.subscribed_stk_tick:
            self.api.quote.unsubscribe(contract, "tick")
            self.subscribed_stk_tick.remove(code)

def unsubscribe_all_stk_tick(self):
    for code in self.subscribed_stk_tick:
        contract = self.api.Contracts.Stocks[code]
        if contract is not None:
            self.api.quote.unsubscribe(contract, "tick")
    self.subscribed_stk_tick.clear()
```

`subscribe_stk_tick` `subscribed_stk_tick` Shioaji `subscribe` `subscribed_stk_tick` `Set`

**`QuoteManager` ticks**

```
def __init__(self, api: sj.Shioaji):
    # skip
    self.df_stk: pl.DataFrame = pl.DataFrame(
        [],
        schema=[
            ("datetime", pl.Datetime),
            ("code", pl.Utf8),
            ("price", pl.Float64),
            ("volume", pl.Int64),
            ("tick_type", pl.Int8),
        ],
    )

def get_df_stk(self) -> pl.DataFrame:
    poped_ticks, self.ticks_stk_v1 = self.ticks_stk_v1, []
    if poped_ticks:
        df = pl.DataFrame([tick.to_dict() for tick in poped_ticks]).select(
            pl.col("datetime", "code"),
            pl.col("close").cast(pl.Float64).alias("price"),
            pl.col("volume").cast(pl.Int64),
            pl.col("tick_type").cast(pl.Int8),
        )
        self.df_stk = self.df_stk.vstack(df)
    return self.df_stk
```

`__init__` `df_stk` Polars DataFrame tick `get_df_stk` `ticks_stk_v1` Polars DataFrame DataFrame

df_stk

**`QuoteManager` ticks K**

```
def get_df_stk_kbar(
    self, unit: str = "1m", exprs: List[pl.Expr] = []
) -> pl.DataFrame:
    df = self.get_df_stk()
    df = df.group_by(
        pl.col("datetime").dt.truncate(unit),
        pl.col("code"),
        maintain_order=True,
    ).agg(
        pl.col("price").first().alias("open"),
        pl.col("price").max().alias("high"),
        pl.col("price").min().alias("low"),
        pl.col("price").last().alias("close"),
        pl.col("volume").sum().alias("volume"),
    )
    if exprs:
        df = df.with_columns(exprs)
    return df
```

`get_df_stk_kbar` `get_df_stk` Ticks DataFrame code truncate datetime DataFrame DataFrame K `exprs` truncate `1m` 1 5 K `5m` 1 K `1h` truncate API

```
import polars as pl
import polars_talib as plta

quote_manager.get_df_stk_kbar("5m", [
    pl.col("close").ta.ema(5).over("code").fill_nan(None).alias("ema5"),
    plta.macd(pl.col("close"), 12, 26, 9).over("code").struct.field("macd").fill_nan(None),
])
```

polars_ta expression K `ema` `macd` polars_ta_extension

polars_ta expression `over("code")` DataFrame
`over` partition `alias` `ema5` `macd` struct struct macd

polars expression

```
def fetch_ticks(self, contract: BaseContract) -> pl.DataFrame:
    code = contract.code
    ticks = self.api.ticks(contract)
    df = pl.DataFrame(ticks.dict()).select(
        pl.from_epoch("ts", time_unit="ns").dt.cast_time_unit("us").alias("datetime"),
        pl.lit(code).alias("code"),
        pl.col("close").alias("price"),
        pl.col("volume").cast(pl.Int64),
        pl.col("tick_type").cast(pl.Int8),
    )
    return df

def subscribe_stk_tick(self, codes: List[str], recover: bool = False):
    for code in codes:
        # skop
        if recover:
            df = self.fetch_ticks(contract)
            if not df.is_empty():
                code_ticks = [t for t in self.ticks_stk_v1 if t.code == code]
                if code_ticks:
                    t_first = code_ticks[0].datetime
                    df = df.filter(pl.col("datetime") < t_first)
                    self.df_stk = self.df_stk.vstack(df)
                else:
                    self.df_stk = self.df_stk.vstack(df)
```

api tick `fetch_ticks` tick `df_stk`

jupyter lab

QuoteManager quote.py

jupyter notebook quote_manager_usage

# 5.

1.0 1.0

## 5.1 Shioaji

`backend`

```
import shioaji as sj
sj.Shioaji?
```

**Out**

**version>=1.0** **version<1.0**

```
Init signature:
sj.Shioaji(
    simulation: bool = False,
    proxies: Dict[str, str] = {},
    currency: str = 'NTD',
)
Docstring:
shioaji api

Functions:
    login
    logout
    activate_ca
    list_accounts
    set_default_account
    get_account_margin
    get_account_openposition
    get_account_settle_profitloss
    get_stock_account_funds
    get_stock_account_unreal_profitloss
    get_stock_account_real_profitloss
    place_order
    update_order
    update_status
    list_trades

Objects:
    Quote
    Contracts
    Order
Init docstring:
initialize Shioaji to start trading

Args:
    simulation (bool):
        - False: to trading on real market (just use your Sinopac account to start trading)
        - True: become simulation account(need to contract as to open simulation account)
    proxies (dict): specific the proxies of your https
        ex: {'https': 'your-proxy-url'}
    currency (str): {NTX, USX, NTD, USD, HKD, EUR, JPY, GBP}
        set the default currency for display
```

```
Init signature:
sj.Shioaji(
    backend: str = 'http',
    simulation: bool = False,
    proxies: Dict[str, str] = {},
    currency: str = 'NTD',
)
Docstring:
shioaji api

Functions:
    login
    activate_ca
    list_accounts
    set_default_account
    get_account_margin
    get_account_openposition
    get_account_settle_profitloss
    get_stock_account_funds
    get_stock_account_unreal_profitloss
    get_stock_account_real_profitloss
    place_order
    update_order
    update_status
    list_trades

Objects:
    Quote
    Contracts
    Order
Init docstring:
initialize Shioaji to start trading

Args:
    backend (str): {http, socket}
        use http or socket as backend currently only support http, async socket backend coming soon.
    simulation (bool):
        - False: to trading on real market (just use your Sinopac account to start trading)
        - True: become simulation account(need to contract as to open simulation account)
    proxies (dict): specific the proxies of your https
        ex: {'https': 'your-proxy-url'}
    currency (str): {NTX, USX, NTD, USD, HKD, EUR, JPY, GBP}
        set the default currency for display
```

## 5.2

`person_id` `passwd` `api_key` `secret_key` Token API Key

In

**version>=1.0** **version<1.0**

```
import shioaji as sj
api = sj.Shioaji()
api.login(
    api_key="YOUR_API_KEY",
    secret_key="YOUR_SECRET_KEY"
)

import shioaji as sj
api = sj.Shioaji()
api.login(
    person_id="YOUR_PERSON_ID",
    passwd="YOUR_PASSWORD",
)
```

Out

```
[
    FutureAccount(person_id='', broker_id='', account_id='', signed=True, username=''),
    StockAccount(person_id='', broker_id='', account_id='', signed=True, username='')
]
```

## 5.3

`TFTStockOrder` `StockOrder`

**StockOrder**

verion>=1.0 verion<1.0

```
>> sj.order.StockOrder?

Init signature:
sj.order.StockOrder(
    *,
    action: shioaji.constant.Action,
    price: Union[pydantic.types.StrictInt, float],
    quantity: shioaji.order.ConstrainedIntValue,
    id: str = '',
    seqno: str = '',
    ordno: str = '',
    account: shioaji.account.Account = None,
    custom_field: shioaji.order.ConstrainedStrValue = '',
    ca: str = '',
    price_type: shioaji.constant.StockPriceType,
    order_type: shioaji.constant.OrderType,
    order_lot: shioaji.constant.StockOrderLot = <StockOrderLot.Common: 'Common'>,
    order_cond: shioaji.constant.StockOrderCond = <StockOrderCond.Cash: 'Cash'>,
    daytrade_short: bool = False,
) -> None

>> sj.order.TFTStockOrder?

Init signature:
sj.order.TFTStockOrder(
    *,
    action: shioaji.constant.Action,
    price: Union[pydantic.types.StrictInt, float],
    quantity: shioaji.order.ConstrainedIntValue,
    id: str = '',
    seqno: str = '',
    ordno: str = '',
    account: shioaji.account.Account = None,
    custom_field: shioaji.order.ConstrainedStrValue = '',
    ca: str = '',
    price_type: shioaji.constant.TFTStockPriceType,
    order_type: shioaji.constant.TFTOrderType,
    order_lot: shioaji.constant.TFTStockOrderLot = <TFTStockOrderLot.Common: 'Common'>,
    order_cond: shioaji.constant.StockOrderCond = <StockOrderCond.Cash: 'Cash'>,
    first_sell: shioaji.constant.StockFirstSell = <StockFirstSell.No: 'false'>,
) -> None
```

## 5.3.1

- `TFTStockPriceType` `StockPriceType`
- `TFTOrderType` `OrderType`
- `TFTStockOrderLot` `StockOrderLot`
- `first_sell` `daytrade_short` `Bool`

**Order**

version>=1.0 version<1.0

```
order = api.Order(
    price=12,
    quantity=1,
    action=sj.constant.Action.Sell,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    order_lot=sj.constant.StockOrderLot.Common,
    daytrade_short=True,
    custom_field="test",
    account=api.stock_account
)

order = api.Order(
    price=12,
    quantity=1,
    action=sj.constant.Action.Sell,
    price_type=sj.constant.TFTStockPriceType.LMT,
    order_type=sj.constant.TFTOrderType.ROD,
    order_lot=sj.constant.TFTStockOrderLot.Common,
    first_sell=sj.constant.StockFirstSell.Yes,
    custom_field="test",
    account=api.stock_account
)
```

## 5.3.2

`TFTOrder` `StockOrder`

**Order Callback**

**version>=1.0** **version<1.0**

```
OrderState.StockOrder {
    'operation': {
        'op_type': 'New',
        'op_code': '00',
        'op_msg': ''
    },
    'order': {
        'id': 'c21b876d',
        'seqno': '429832',
        'ordno': 'W2892',
        'action': 'Buy',
        'price': 12.0,
        'quantity': 10,
        'order_cond': 'Cash',
        'order_lot': 'Common',
        'custom_field': 'test',
        'order_type': 'ROD',
        'price_type': 'LMT'
    },
    'status': {
        'id': 'c21b876d',
        'exchange_ts': 1583828972,
        'modified_price': 0,
        'cancel_quantity': 0,
        'web_id': '137'
    },
    'contract': {
        'security_type': 'STK',
        'exchange': 'TSE',
        'code': '2890',
        'symbol': '',
        'name': '',
        'currency': 'TWD'
    }
}
```

```
OrderState.TFTOrder {
    'operation': {
        'op_type': 'New',
        'op_code': '00',
        'op_msg': ''
    },
    'order': {
        'id': 'c21b876d',
        'seqno': '429832',
        'ordno': 'W2892',
        'action': 'Buy',
        'price': 12.0,
        'quantity': 10,
        'order_cond': 'Cash',
        'order_lot': 'Common',
        'custom_field': 'test',
        'order_type': 'ROD',
        'price_type': 'LMT'
    },
    'status': {
        'id': 'c21b876d',
        'exchange_ts': 1583828972,
        'modified_price': 0,
        'cancel_quantity': 0,
        'web_id': '137'
    },
    'contract': {
        'security_type': 'STK',
        'exchange': 'TSE',
        'code': '2890',
        'symbol': '',
        'name': '',
        'currency': 'TWD'
    }
}
```

## 5.3.3

`TFTDeal` `StockDeal`

**Deal Callback**

version>=1.0 version<1.0

```
OrderState.StockDeal {
    'trade_id': '12ab3456',
    'exchange_seq': '123456',
    'broker_id': 'your_broker_id',
    'account_id': 'your_account_id',
    'action': <Action.Buy: 'Buy'>,
    'code': '2890',
    'order_cond': <StockOrderCond.Cash: 'Cash'>,
    'order_lot': <TFTStockOrderLot.Common: 'Common'>,
    'price': 12,
    'quantity': 10,
    'web_id': '137',
    'custom_field': 'test',
    'ts': 1583828972
}

OrderState.TFTDeal {
    'trade_id': '12ab3456',
    'exchange_seq': '123456',
    'broker_id': 'your_broker_id',
    'account_id': 'your_account_id',
    'action': <Action.Buy: 'Buy'>,
    'code': '2890',
    'order_cond': <StockOrderCond.Cash: 'Cash'>,
    'order_lot': <TFTStockOrderLot.Common: 'Common'>,
    'price': 12,
    'quantity': 10,
    'web_id': '137',
    'custom_field': 'test',
    'ts': 1583828972
}
```

## 5.4

**FuturesOrder**

verion>=1.0 verion<1.0

```
>> sj.order.FuturesOrder?

Init signature:
sj.order.FuturesOrder(
    *,
    action: shioaji.constant.Action,
    price: Union[pydantic.types.StrictInt, float],
    quantity: shioaji.order.ConstrainedIntValue,
    id: str = '',
    seqno: str = '',
    ordno: str = '',
    account: shioaji.account.Account = None,
    custom_field: shioaji.order.ConstrainedStrValue = '',
    ca: str = '',
    price_type: shioaji.constant.FuturesPriceType,
    order_type: shioaji.constant.OrderType,
    octype: shioaji.constant.FuturesOCType = <FuturesOCType.Auto: 'Auto'>,
) -> None

>> sj.order.FuturesOrder?

Init signature:
sj.order.FuturesOrder(
    *,
    action: shioaji.constant.Action,
    price: Union[pydantic.types.StrictInt, float],
    quantity: shioaji.order.ConstrainedIntValue,
    id: str = '',
    seqno: str = '',
    ordno: str = '',
    account: shioaji.account.Account = None,
    custom_field: shioaji.order.ConstrainedStrValue = '',
    ca: str = '',
    price_type: shioaji.constant.FuturesPriceType,
    order_type: shioaji.constant.FuturesOrderType,
    octype: shioaji.constant.FuturesOCType = <FuturesOCType.Auto: 'Auto'>,
) -> None
```

### 5.4.1

`FuturesOrderType` `OrderType`

**Order**

**verion>=1.0** **verion<1.0**

```
order = api.Order(
    action=sj.constant.Action.Buy,
    price=100,
    quantity=1,
    price_type=sj.constant.FuturesPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    octype=sj.constant.FuturesOCType.Auto,
    account=api.futopt_account
)

order = api.Order(
    action=sj.constant.Action.Buy,
    price=100,
    quantity=1,
    price_type=sj.constant.FuturesPriceType.LMT,
    order_type=sj.constant.FuturesOrderType.ROD,
    octype=sj.constant.FuturesOCType.Auto,
    account=api.futopt_account
)
```

## 5.4.2

`FOrder` `FuturesOrder`

**Order Event**

**version>=1.0** **version<1.0**

```
OrderState.FuturesOrder {
    'operation': {
        'op_type': 'New',
        'op_code': '00',
        'op_msg': ''
    },
    'order': {
        'id': '02c347f7',
        'seqno': '956201',
        'ordno': 'kY00H',
        'action': 'Sell',
        'price': 17760.0,
        'quantity': 1,
        'order_cond': None,
        'order_type': 'ROD',
        'price_type': 'LMT',
        'market_type': 'Night',
        'oc_type': 'New',
        'subaccount': ''
    },
    'status': {
        'id': '02c347f7',
        'exchange_ts': 1625729890,
        'modified_price': 0.0,
        'cancel_quantity': 0,
        "web_id": "P"
    },
    'contract': {
        'security_type': 'FUT',
        'code': 'TXF',
        'exchange': 'TIM',
        'delivery_month': '202107',
        'strike_price': 0.0,
        'option_right': 'Future'
    }
}
```

```
OrderState.FOrder {
    'operation': {
        'op_type': 'New',
        'op_code': '00',
        'op_msg': ''
    },
    'order': {
        'id': '02c347f7',
        'seqno': '956201',
        'ordno': 'kY00H',
        'action': 'Sell',
        'price': 17760.0,
        'quantity': 1,
        'order_cond': None,
        'order_type': 'ROD',
        'price_type': 'LMT',
        'market_type': 'Night',
        'oc_type': 'New',
        'subaccount': ''
    },
    'status': {
        'id': '02c347f7',
        'exchange_ts': 1625729890,
        'modified_price': 0.0,
        'cancel_quantity': 0,
        "web_id": "P"
    },
    'contract': {
        'security_type': 'FUT',
        'code': 'TXF',
        'exchange': 'TIM',
        'delivery_month': '202107',
        'strike_price': 0.0,
        'option_right': 'Future'
    }
}
```

## 5.4.3

`FDeal` `FuturesDeal`

**Deal Event**

**version>=1.0** **version<1.0**

```
OrderState.FuturesDeal {
    "trade_id":"02c347f7",
    "seqno":"956344",
    "ordno":"ky00N11O",
    "exchange_seq":"a0000060",
    "broker_id":"F002000",
    "account_id":"9104000",
    "action":"Sell",
    "code":"TXF",
    "price":17650.0,
    "quantity":4,
    "subaccount":"",
    "security_type":"FUT",
    "delivery_month":"202107",
    "strike_price":0.0,
    "option_right":"Future",
    "market_type":"Day",
    "ts":1625800369
}
```

```
OrderState.FDeal {
    "trade_id":"02c347f7",
    "seqno":"956344",
    "ordno":"ky00N11O",
    "exchange_seq":"a0000060",
    "broker_id":"F002000",
    "account_id":"9104000",
    "action":"Sell",
    "code":"TXF",
    "price":17650.0,
    "quantity":4,
    "subaccount":"",
    "security_type":"FUT",
    "delivery_month":"202107",
    "strike_price":0.0,
    "option_right":"Future",
    "market_type":"Day",
    "ts":1625800369
}
```

# 5.5

**>=1.1** **QuoteVersion.v0** **QuoteVersion.v1**

## 5.5.1 Callback

**Tick**

**decorator**

**QuoteVersion.v1** **QuoteVersion.v0**

```
from shioaji import TickSTKv1, Exchange

@api.on_tick_stk_v1()
def quote_callback(exchange: Exchange, tick:TickSTKv1):
    print(f"Exchange: {exchange}, Tick: {tick}")

@api.quote.on_quote
def quote_callback(topic: str, quote: dict):
    print(f"Topic: {topic}, Quote: {quote}")
```

**QuoteVersion.v1** **QuoteVersion.v0**

```
from shioaji import TickSTKv1, Exchange

def quote_callback(exchange: Exchange, tick:TickSTKv1):
    print(f"Exchange: {exchange}, Tick: {tick}")

api.quote.set_on_tick_stk_v1_callback(quote_callback)

def quote_callback(topic: str, quote: dict):
    print(f"Topic: {topic}, Quote: {quote}")

api.quote.set_quote_callback(quote_callback)
```

**Out**

**QuoteVersion.v1** **QuoteVersion.v0**

```
Exchange: Exchange.TSE, Tick: Tick(code='2330', datetime=datetime.datetime(2021, 7, 2, 13, 16, 35, 92970), open=Decimal('590'), avg_price=Decimal('589.05'),
close=Decimal('590'), high=Decimal('593'), low=Decimal('587'), amount=Decimal('590000'), total_amount=Decimal('8540101000'), volume=1, total_volume=14498,
tick_type=1, chg_type=4, price_chg=Decimal('-3'), pct_chg=Decimal('-0.505902'), trade_bid_volume=6638, ask_side_total_vol=7860, bid_side_total_cnt=2694,
ask_side_total_cnt=2705, closing_oddlot_shares=0, fixed_trade_vol=0, suspend=0, simtrade=0, intraday_odd=0)

Topic: MKT/*/TSE/2330, Quote: {'AmountSum': [4739351000.0], 'Close': [596.0], 'Date': '2021/03/30', 'TickType': [2], 'Time': '10:01:33.349431', 'VolSum': [7932],
'Volume': [1]}
```

### BidAsk

**decorator**

**QuoteVersion.v1** **QuoteVersion.v0**

```
from shioaji import BidAskSTKv1, Exchange

@api.on_bidask_stk_v1()
def quote_callback(exchange: Exchange, bidask:BidAskSTKv1):
    print(f"Exchange: {exchange}, BidAsk: {bidask}")

@api.quote.on_quote
def quote_callback(topic: str, quote: dict):
    print(f"Topic: {topic}, Quote: {quote}")
```

**QuoteVersion.v1** **QuoteVersion.v0**

```
from shioaji import BidAskSTKv1, Exchange

def quote_callback(exchange: Exchange, bidask:BidAskSTKv1):
    print(f"Exchange: {exchange}, BidAsk: {bidask}")

api.quote.set_on_bidask_stk_v1_callback(quote_callback)

def quote_callback(topic: str, quote: dict):
    print(f"Topic: {topic}, Quote: {quote}")

api.quote.set_quote_callback(quote_callback)
```

Out

QuoteVersion.v1 QuoteVersion.v0

```
Exchange: Exchange.TSE, BidAsk: BidAsk(code='2330', datetime=datetime.datetime(2021, 7, 2, 13, 17, 29, 726428), bid_price=[Decimal('589'), Decimal('588'),
Decimal('587'), Decimal('586'), Decimal('585')], bid_volume=[223, 761, 1003, 809, 1274], diff_bid_vol=[0, 0, 0, 0, 0], ask_price=[Decimal('590'), Decimal('591'),
Decimal('592'), Decimal('593'), Decimal('594')], ask_volume=[304, 232, 183, 242, 131], diff_ask_vol=[1, 0, 0, 0, 0], suspend=0, simtrade=0, intraday_odd=0)

Topic: QUT/idcdmzpcr01/TSE/2330, Quote: {'AskPrice': [590.0, 591.0, 592.0, 593.0, 594.0], 'AskVolume': [303, 232, 183, 242, 131], 'BidPrice': [589.0, 588.0,
587.0, 586.0, 585.0], 'BidVolume': [224, 762, 1003, 809, 1274], 'Date': '2021/07/02', 'Time': '13:17:26.391840'}
```

## 5.6

API

```
1. get_account_margin
2. get_account_openposition
3. get_account_settle_profitloss
```

```
1. margin
2. list_positions( api.futopt_account )
3. list_profit_loss( api.futopt_account )
4. list_profit_loss_detail( api.futopt_account )
5. list_profit_loss_summary( api.futopt_account )
```

API

GITHUB

github

# 6.

## 6.0.1

**(MKT) (MKP)**

```
order = api.Order(
    action=sj.constant.Action.Buy,
    price=0, # MKT, MKP will not use price parameter
    quantity=1,
    price_type='MKP', # change to MKT or MKP
    order_type='IOC', # MKT, MKP only accecpt IOC order
    octype=sj.constant.FuturesOCType.Auto,
    account=api.futopt_account
)
```

**( ) ROD**

First, we need to know the `limit up(limit down)` price of the security. Just take a look at the `api.Contracts`, you will find the information you want.

**In**

```
api.Contracts.Stocks.TSE['TSE2330']
```

**Out**

```
Stock(
    exchange=<Exchange.TSE: 'TSE'>,
    code='2330',
    symbol='TSE2330',
    name='    ',
    category='24',
    unit=1000,
    limit_up=653.0,
    limit_down=535.0,
    reference=594.0,
    update_date='2021/08/27',
    margin_trading_balance=6565,
    short_selling_balance=365,
    day_trade=<DayTrade.Yes: 'Yes'>
)
```

Example place LMT and ROD order at limit up price.

**In**

```
contract = api.Contracts.Stocks.TSE['TSE2330']
price = contract.limit_up
order = api.Order(
    action=sj.constant.Action.Buy,
    price=price,
    quantity=1,
    price_type='LMT',
    order_type='ROD',
    order_lot=sj.constant.StockOrderLot.Common,
    account=api.stock_account
)
```

## 6.0.2

If your code something like this, and possibly run code on cmd/terminal with `python stream.py`. Then you definitely won't get any additional ticks, since the python program has already terminated.

**version>=1.0** **version<1.0**

```
import shioaji as sj

api = sj.Shioaji(simulation=True)
api.login('YOUR_API_KEY', 'YOUR_SECRET_KEY')
api.quote.subscribe(
    api.Contracts.Stocks["2330"],
    quote_type = sj.constant.QuoteType.Tick
)

# stream.py
import shioaji as sj

api = sj.Shioaji(simulation=True)
api.login('YOUR_PERSON_ID', '2222')
api.quote.subscribe(
    api.Contracts.Stocks["2330"],
    quote_type = sj.constant.QuoteType.Tick
)
```

If you wish your python program to survive, please modify you python script as below.

**version>=1.0** **version<1.0**

```
# stream.py
import shioaji as sj
from threading import Event

api = sj.Shioaji(simulation=True)
api.login('YOUR_API_KEY', 'YOUR_SECRET_KEY')
api.quote.subscribe(
    api.Contracts.Stocks["2330"],
    quote_type = sj.constant.QuoteType.Tick
)

Event().wait()

# stream.py
import shioaji as sj
from threading import Event

api = sj.Shioaji(simulation=True)
api.login('YOUR_PERSON_ID', '2222')
api.quote.subscribe(
    api.Contracts.Stocks["2330"],
    quote_type = sj.constant.QuoteType.Tick
)

Event().wait()
```

## 6.0.3

**Account not acceptable**

```
-    [  ](https://sinotrade.github.io/zh_TW/tutor/prepare/terms/#_1) [API  ](https://sinotrade.github.io/zh_TW/tutor/prepare/terms/#api)
- [`update_status`](../tutor/order/UpdateStatus)
```

**shioaji.log**

Please add environment variable before import shioaji. `(version >= 0.3.3.dev0)`

linux or Mac OS:

```
export SJ_LOG_PATH=/path/to/shioaji.log
```

windows:

```
set SJ_LOG_PATH=C:\path\to\shioaji.log
```

**contracts**

Please add environment variable before import shioaji. `(version >= 0.3.4.dev2)`

linux or Mac OS:

```
export SJ_CONTRACTS_PATH=MY_PATH
```

windows:

```
set SJ_CONTRACTS_PATH=MY_PATH
```

python:

```
os.environ["SJ_CONTRACTS_PATH"]=MY_PATH
```

**3**

** Note that you only have 2 chances to unlock your account online in a day. **

** We've migrate QA site to Shioaji Forum **

# 7.

## 7.1 version: 1.2.5 (2024-10-01)

- feat: refactor expire time of CA

commit_id: 6621685a

⚡
release_at: 2024-10-01 02:25:01.723

## 7.2 version: 1.2.4 (2024-08-28)

- feat: support py3.12

commit_id: a287f56c

⚡
release_at: 2024-08-28 16:00:00.000

## 7.3 version: 1.2.3 (2024-03-06)

- feat: change default site to bc
- feat: pysolace upgrade 0.9.40(solclient 7.28.0.4)
- feat: support apple silicon chip

commit_id: 8096bbac

⚡
release_at: 2024-03-06 16:00:00.000

## 7.4 version: 1.2.2 (2024-01-09)

- fix: remove column of profitloss in future

commit_id: ca973a81

⚡
release_at: 2024-01-09 02:27:31.383

## 7.5 version: 1.2.1 (2023-12-22)

- fix: windows inject dll issue

commit_id: 2a413848

⚡
release_at: 2023-12-22 01:19:17.043

## 7.6 version: 1.2.0 (2023-12-20)

- feat: vpn
- feat: rust version ca

- refactor: test report flow

commit_id: 856f39ea

⚡
release_at: 2023-12-20 16:00:00.000

## 7.7 version: 1.1.13 (2023-11-01)

feat: impl ca.get_sign on Darwin

commit_id: 729f058e

⚡
release_at: 2023-11-01 05:36:29.553

## 7.8 version: 1.1.12 (2023-08-22)

- feat: usage add limit and available byte info

commit_id: cf5e4628

⚡
release_at: 2023-08-22 16:00:00.000

## 7.9 version: 1.1.11 (2023-08-04)

- fix: custom_field in validator for only support number and alphabet
- fix: pydantic v2 trade issue
- fix: pydantic v2 contracts cache issue

commit_id: cc1da47e

⚡
release_at: 2023-08-04 08:00:37.000

## 7.10 version: 1.1.10 (2023-07-23)

- feat: profit loss detail support unit

commit_id: a62d1f6a

⚡
release_at: 2023-07-23 16:00:00.000

## 7.11 version: 1.1.9 (2023-07-20)

**yanked**

commit_id: f9f03cff

⚡
release_at: 2023-07-20 07:43:46.000

## 7.12 version: 1.1.8 (2023-07-18)

- feat: query usage
- feat: profit_loss support unit
- feat: support pydantic v2

commit_id: 3dc8568e

⚡
release_at: 2023-07-18 09:08:42.000

## 7.13 version: 1.1.7 (2023-07-18)

**yanked**

commit_id: fb490a9a

⚡
release_at: 2023-07-18 07:02:20.000

## 7.14 version: 1.1.6 (2023-06-19)

- feat: solace reconnect with sub 2500 user

commit_id: eeaeeb5f

⚡
release_at: 2023-06-19 16:00:00.000

## 7.15 version: 1.1.5 (2023-06-07)

- fix: simulation sign check
- fix: handle pickle load error

commit_id: 1def691c

⚡
release_at: 2023-06-07 16:00:00.000

# 8.

- :

  **30** **API**

| | |
|---|---|
| 0 | 500MB |
| 1 - 1 | 2GB |
| > 1 | 10GB |

- :

  **30** **API**

| | |
|---|---|
| 0 | 500MB |
| 1 - 1000 / 4000 | 2GB |
| > 1000 / 4000 | 10G |

```
api.usage()
```

**Out**

```
UsageStatus(connections=1, bytes=41343260, limit_bytes=2147483648, remaining_bytes=2106140388)
```

```
connection:
bytes:
limit_bytes:
remaining_bytes:
```

- :
  `credit_enquire`, `short_stock_sources`, `snapshots`, `ticks`, `kbars`
- 5 50
- `ticks` 10
- `kbars` 270
- :
  `list_profit_loss_detail`,`account_balance`, `list_settlements`, `list_profit_loss`, `list_positions`, `margin`
  5 25
- :
  `place_order`, `update_status`, `update_qty`, `update_price`, `cancel_order`
  10 250
- :
  `api.subscribe()` 200
- :
  `person_id` 5 : `api.login()`
- :
  `api.login()` 1000

**Warn**

- (ticks snapshots kbars)
-
- IP ID
- ID Shioaji