KungFu V2.2.0 API 文档

快速上手

版本更新说明

2.2.0

更新时间：2020.04.23

- 优化前后端通信方式
- 添加实时延迟统计功能
- 支持极速模式运行
- 调整账目持仓统计方式

2.1.0

更新时间：2019.11.15

- 添加策略获取账户持仓接口
- 区分股票与期货持仓数据结构
- 添加bar订阅与回调

2.0.0

更新时间：2019.10.8

- 策略示例
- 函数定义
- 数据结构定义
- 常量定义

1.0.0

首次发布：2017.11.15

策略接口调整对照表

功能描述	2.1	2.2
可用资金	Book.avail	Book.asset.avail
保证金	Book.margin	Book.asset.margin
市值	Book.market_value	Book.asset.market_value
初始权益	Book.initial_equity	Book.asset.initial_equity
动态权益	Book.dynamic_equity	Book.asset.dynamic_equity
静态权益	Book.static_equity	Book.asset.static_equity
已实现盈亏	Book.realized_pnl	Book.asset.realized_pnl
未实现盈亏	Book.unrealized_pnl	Book.asset.unrealized_pnl
2.1持仓列表/2.2多头持仓列表	Book.positions	Book.long_positions
2.1持仓列表/2.2空头持仓列表	Book.positions	Book.short_positions
获取持仓明细	Book.get_position(instrument_id, exchange_id,direction)	Book.get_position(direction, exchange_id,instrument_id)
自动订阅历史上持有过合约		context.hold_book()
使用策略持仓，默认不调用使用账户持仓		context.hold_positions()

策略逻辑简介

功夫系统上的策略在策略连接行情交易柜台，发送订阅请求以后，通过以下回调函数给用户发送消息，用户在不同的回调函数中调用功能函数实现获取行情，下单，时间回调等逻辑。详细数据定义和接口函数定义可查看后文，以下为一个策略示例

```
# -*- coding: UTF-8 -*-
import kungfu.yijinjing.time as kft
from kungfu.wingchun.constants import *


# 定义股票账号、柜台、交易所以及需要订阅的标的
SOURCE = "tora"
EXCHANGE= Exchange.SSE
ACCOUNT= "15040900"
tickers= ["600000","600001"]
VOLUME = 2

# 定义期货账号、柜台、交易所以及需要订阅的标的
# SOURCE = "ctp"
# EXCHANGE_FUTURE = Exchange.SHFE
# ACCOUNT_FUTURE = "089270"
```

```
# tickers_FUTURE = ["rb2001","rb2003"]
# VOLUME = 2

# 定义期权账号、柜台、交易所以及需要订阅的标的，目前期权只支持 ToraOption 柜台
# SOURCE = "ToraOption"
# EXCHANGE_OPTION = Exchange.SSE
# ACCOUNT_OPTION = "15040902"
# tickers_OPTION = ["510050","510300"]
# VOLUME = 2


# 启动前回调，添加交易账户，订阅行情，策略初始化计算等
def pre_start(context):
    # 添加股票柜台、账户以及配置账户初始资金
    context.add_account(SOURCE, ACCOUNT, 100000.0)
    # 添加期货柜台、账户以及配置账户初始资金
    # context.add_account(SOURCE_FUTURE, ACCOUNT_FUTURE, 100000.0)
    # 添加期权柜台、账户以及配置账户初始资金
    # context.add_account(SOURCE_OPTION, ACCOUNT_OPTION, 100000.0)

    # 订阅股票行情
    context.subscribe(SOURCE, tickers, EXCHANGE)
    # 订阅期货行情
    # context.subscribe(SOURCE_FUTURE, tickers_FUTURE, EXCHANGE_FUTURE)
    # 订阅期权行情
    # context.subscribe(SOURCE_OPTION, tickers_OPTION, EXCHANGE_OPTION)

    #context.subscribe("bar", tickers, EXCHANGE)
    context.ordered = False

# 启动准备工作完成后回调，策略只能在本函数回调以后才能进行获取持仓和报单
def post_start(context):
    context.log.warning("post_start")
    log_book(context,None)

# 收到快照行情时回调，行情信息通过quote对象获取
def on_quote(context, quote):
    # context.log.info("[on_quote] {}".format(quote))
    if quote.instrument_id in tickers:
        # 如果没有报单则报单
        if not context.ordered:
            order_id = context.insert_order(quote.instrument_id, EXCHANGE,
ACCOUNT, quote.last_price, VOLUME, PriceType.Limit, Side.Buy, Offset.Open)
            # order_id = context.insert_order(quote.instrument_id, EXCHANGE,
ACCOUNT, quote.last_price, VOLUME, PriceType.Any, Side.Buy, Offset.Open)#上期所
不支持市价单
            if order_id > 0:
                context.ordered = True
```

```
                context.log.info("[order] (rid){} (ticker){}".format(order_id,
quote.instrument_id))
                # 通过添加时间回调，在三秒以后撤单
                context.add_timer(context.now() + 3 * 1000000000, lambda ctx,
event: cancel_order(ctx, order_id))

def on_transaction(context, transaction):
    context.log.info("[on_transaction] {}".format(transaction))

def on_entrust(context, entrust):
    context.log.info("[on_entrust] {}".format(entrust))

# 收到k线行情时回调，行情信息通过bar对象获取
def on_bar(context, bar):
    context.log.warning("[on_bar] {}".format(bar))


# 收到订单状态回报时回调
def on_order(context, order):
    context.log.info("[on_order] {}".format(order))


# 收到成交信息回报时回调
def on_trade(context, trade):
    context.log.info("[on_trade] {}".format(trade))


# 策略释放资源前回调，仍然可以获取持仓和报单
def pre_stop(context):
    context.log.info("[befor strategy stop]")


# 策略释放资源后回调
def post_stop(context):
    context.log.info("[befor process stop]")


## 自定义函数
# 自定义持仓和资金日志输出函数
def log_book(context, event):
    context.log.warning("[avail]{}".format(context.book.asset.avail))
    context.log.warning("[acc_avail]
{}".format(context.get_account_book(SOURCE, ACCOUNT).asset.avail))
    book = context.book
    asset = book.asset
    context.log.warning(
        "[strategy capital] (avail){} (margin){} (market_value){}
(initial_equity){} (dynamic_equity){} (static_equity){} (realized_pnl){}
(unrealized_pnl){}".format(
```

```
            asset.avail, asset.margin, asset.market_value,
asset.initial_equity,
            asset.dynamic_equity, asset.static_equity, asset.realized_pnl,
asset.unrealized_pnl))
    book = context.get_account_book(SOURCE, ACCOUNT)
    asset = book.asset
    context.log.warning(
        "[account capital] (avail){} (margin){} (market_value){}
(initial_equity){} (dynamic_equity){} (static_equity){} (realized_pnl){}
(unrealized_pnl){}".format(
            asset.avail, asset.margin, asset.market_value,
asset.initial_equity,
            asset.dynamic_equity, asset.static_equity, asset.realized_pnl,
asset.unrealized_pnl))
    context.logger.warning("acc_pos")
    for key in book.long_positions:
        log_stock_pos(context, book.long_positions[key])
    # log_future_pos(context, pos)
    context.logger.warning("str_pos")
    for key in context.book.long_positions:
        log_stock_pos(context, context.book.long_positions[key])
        # log_future_pos(context, pos)

# 自定义期货持仓数据日志输出函数
def log_future_pos(context, pos):
    context.log.info(
        "(instrument_id){} (instrument_type){} (exchange_id){}(direction){}
(volume){} (yesterday_volume){} (last_price){} (settlement_price){}
(pre_settlement_price){} (avg_open_price){} (position_cost_price){} (margin){}
(position_pnl){} (realized_pnl){} (unrealized_pnl){} ".format(
            pos.instrument_id, pos.instrument_type, pos.exchange_id,
            pos.direction, pos.volume, pos.yesterday_volume, pos.last_price,
pos.settlement_price, pos.pre_settlement_price,
            pos.avg_open_price, pos.position_cost_price, pos.margin,
pos.position_pnl, pos.realized_pnl, pos.unrealized_pnl))

# 自定义股票持仓数据日志输出函数
def log_stock_pos(context, pos):
    context.log.info(
        "(instrument_id){} (instrument_type){} (exchange_id){} (direction){}
(volume){} (yesterday_volume){} (last_price){} (pre_close_price){}
(close_price){} (avg_open_price){} (position_cost_price){} (realized_pnl){}
(unrealized_pnl){} ".format(
            pos.instrument_id, pos.instrument_type, pos.exchange_id,
pos.direction,
            pos.volume, pos.yesterday_volume, pos.last_price,
pos.pre_close_price, pos.close_price, pos.avg_open_price,
            pos.position_cost_price, pos.realized_pnl, pos.unrealized_pnl))
```

```
# 自定义撤单回调函数
def cancel_order(context, order_id):
    action_id = context.cancel_order(order_id)
    if action_id > 0:
        context.log.info("[cancel order] (action_id){} (rid){}
".format(action_id, order_id))
```

函数定义

基本方法

pre_start

> 启动前调用函数，在策略启动前调用，用于完成添加交易账户，订阅行情，策略初始化计算等

参数

参数	类型	说明
context	python对象	策略的全局变量，通过点标记（”.”）来获取其属性。

返回

返回	类型	说明
无	无	无

范例

```
def pre_start(context):
    context.add_account(source, account, 100000.0)
    context.subscribe(source, tickers, exchange)
    context.count = 0
```

post_start

> 启动后调用函数，策略连接上行情交易柜台后调用，本函数回调后，策略可以执行添加时间回调、获取策略持仓、报单等操作

参数

参数	类型	说明
context	python对象	策略的全局变量，通过点标记（”.”）来获取其属性。

返回

返回	类型	说明
无	无	无

范例

```
def post_start(context):
    context.log.info("[log] {}".format("post_start"))
    context.add_timer(context.now() + 1*1000000000, lambda ctx, event:
call(ctx, "test log1", "test log2"))
    context.add_timer(context.now() + 2*1000000000, log_pos)
```

pre_stop

> 退出前方法

参数

参数	类型	说明
context	python对象	策略的全局变量，通过点标记（”.”）来获取其属性。

返回

返回	类型	说明
无	无	无

范例

```
# 退出前函数
def pre_stop(context):
    context.log.info("strategy will stop")
```

post_stop

> 退出前方法

参数

参数	类型	说明
context	python对象	策略的全局变量，通过点标记（"."）来获取其属性。

返回

返回	类型	说明
无	无	无

范例

```
# 退出前函数
def post_stop(context):
    context.log.info("process will stop")
```

on_quote

> 快照行情数据的推送会自动触发该方法的调用。

参数

参数	类型	说明
context	python 对象	策略的全局变量，通过点标记（”.”）来获取其属性。
quote	Quote 对象	快照行情数据。

返回

返回	类型	说明
无	无	无

范例

```
def on_quote(context, quote):
    context.log.info('[on_quote] {}, {}, {}'.format( quote.instrument_id,
quote.last_price, quote.volume))
```

on_transaction

> 逐笔成交行情数据的推送会自动触发该方法的调用。

参数

参数	类型	说明
context	python 对象	策略的全局变量，通过点标记（”.”）来获取其属性。
transaction	Transaction 对象	逐笔成交行情数据。

返回

返回	类型	说明
无	无	无

范例

```
def on_transaction(context, transaction):
    context.log.info("[on_transaction] {}".format(transaction))
```

on_entrust

> 逐笔委托行情数据的推送会自动触发该方法的调用。

参数

参数	类型	说明
context	python 对象	策略的全局变量，通过点标记（”.”）来获取其属性。
entrust	Entrust 对象	逐笔委托行情数据。

返回

返回	类型	说明
无	无	无

范例

```
def on_entrust(context, entrust):
    context.log.info("[on_entrust] {}".format(entrust))
```

on_bar

> BAR行情数据的推送会自动触发该方法的调用

参数

参数	类型	说明
context	python 对象	策略的全局变量，通过点标记（”.”）来获取其属性。
bar	Bar 对象	bar 行情数据

返回

返回	类型	说明
无	无	无

范例

```
def on_bar(context, bar):
    context.log.warning("[on_bar] {}".format(bar))
```

on_order

> 订单信息的更新会自动触发该方法的调用。

参数

参数	类型	说明
context	python 对象	策略的全局变量，通过点标记（”.”）来获取其属性。
order	Order 对象	订单信息更新数据。

返回

返回	类型	说明
无	无	无

范例

```
def on_order(context, order):
    context.log.info('[on_order] {}, {}, {}, {}'.format( order.order_id,
order.status, order.volume, order_volume_left))
```

on_trade

> 策略订单成交信息的更新会自动触发该方法的调用。

参数

参数	类型	说明
context	python 对象	策略的全局变量，通过点标记（”.”）来获取其属性。
trade	Trade 对象	订单信息更新数据。

返回

返回	类型	说明
无	无	无

范例

```
def on_trade(context, trade):
    context.log.info('[on_trade] {}, {}, {}'.format(trade.order_id,
trade.volume, trade.price))
```

行情交易函数

context.add_account

> 添加交易柜台，策略需要先添加账户，才能使用该账户报单

参数

参数	类型	说明
source_id	str	行情柜台ID
account_id	str	账户ID
cash_limit	float	策略首次运行时才设置初始资金

返回

返回	类型	说明
result	bool	

范例

```
# 添加柜台、账户以及初始资金
context.add_account(source_id, account_id, cash_limit)
```

context.suscribe

> 订阅行情

参数

参数	类型	说明
source	str	行情柜台ID
instrument	list	代码列表
exchange_id	Exchange	交易所ID
is_level2	bool	是否Level2

返回

返回	类型	说明
无	无	无

范例

```
# 向source柜台的exchange_id交易所订阅了instruments列表中的合约的行情
context.subscribe(source, instruments, exchange_id, is_level2=False)
```

context.insert_order

> 报单函数

参数

参数	类型	说明
instrument_id	str	合约ID
exchange_id	str	交易所ID
account_id	str	交易账号
limit_price	float	价格
volume	str	数量
price_type	Price_Type 对象	报单类型
side	Side 对象	买卖方向
offset	Offset 对象	开平方向

返回

返回	类型	说明
order_id	long	订单ID

范例

- 通过交易账户acc_1以12.0元的价格买200股浦发银行：

```
context.insert_order("600000", Exchange.SSE, "acc_1", 12.0, 200,
PriceType.Limit, Side.Buy, Offset.Open)
```

- 通过交易账户acc_2以3500元的价格开仓买入2手上期所rb1906合约：

```
context.insert_order("rb1906", Exchange.SHFE, "acc_2", 3500.0, 2,
PriceType.Limit, Side.Buy, Offset.Open)
```

注（期权）：当买卖方向为：Lock（锁仓）、Unlock（解锁）、Exec（行权）、Drop（放弃行权）时，设定的price（委托价）以及offset（开平方向）都不生效

context.cancel_order

> 撤单函数

参数

参数	类型	说明
order_id	long	订单ID

返回

返回	类型	说明
action_id	long	订单操作

范例

```
# 通过context.insert_order函数进行下单，同时用order_id记录下单的订单ID号，下单完成后，为了保证后续可能会出现的撤单操作能够正常执行，这里需要等待一段时间
order_id = context.insert_order(quote.instrument_id, exchange, account, quote.last_price - 10, volume, PriceType.Limit, Side.Buy, Offset.Open)
            if order_id > 0:
                context.log.info("[order] (rid){} (ticker){}".format(order_id, quote.instrument_id))
                context.add_timer(context.now() + 1*1000000000, lambda ctx, event: cancel_order(ctx, order_id))
                context.order_id  = order_id
# 将order_id传入cancel_order进行撤单操作，同时打印相关日志
def cancel_order(context, order_id):
    action_id =  context.cancel_order(order_id)
    if action_id > 0:
        context.log.info("[cancel order] (action_id){} (rid){}".format(action_id, order_id))
```

投资组合相关功能

盈亏及持仓

> 功夫系统支持实时维护策略收益及持仓及对应的历史记录，针对不同的应用场景，提供共计四种不同的维护收益及持仓的模式。对于任一策略，具体采用的模式由两个 API 决定：context.hold_book() 及 context.hold_positions()，使用者需要在策略的 pre_start() 方法里决定是否调用这两个方法，系统在 pre_start() 处理完成时会根据是否调用这两个方法对应出的共计四种状态来设置维护收益及持仓的结果。

context.hold_book()

> 保持策略运行历史上的交易过的标的。缺省设置即没有调用此方法时，系统只会维护当前策略代码中通过 subscribe 方法订阅过的标的；当调用此方法后，系统会在策略启动后，根据该同名策略在历史上的交易情况，构造一份包含所有该同名策略所交易过标的，及当前策略代码中通过 subscribe 订阅的标的的账目。注意此方法仅影响标的列表，对于每个标的的具体持仓数值，是由 hold_positions() 方法来决定。

范例

```
# 历史曾执行过订阅某些标的
# context.subscribe(source, ['600000', '600001'])
# 当前代码中重置了 subscribe，且没有调用 context.hold_book()，则该策略只会收到新订阅的标的行情，且账目收益及持仓中只包含新订阅的标的
context.subscribe(source, ['600002', '600003'])
# 如果调用 hold_book，则该策略订阅行情列表中会自动包含历史记录中有的标的，且账目收益及持仓中也会包含对应标的的数据
context.hold_book()
```

context.hold_positions()

> 保持账目中每一标的的历史持仓。缺省设置即没有调用此方法时，系统会通过同步柜台查询到的持仓数据来构建策略账目，每次策略启动后，账目中所有标的的持仓都会同步为最新的柜台账户对应的持仓；当调用此方法后，系统会使用功夫内部记录的历史数据来恢复策略的账目持仓。缺省设置保证了策略账目中的持仓数据是绝对准确的，但无法反映功夫运行期间内的策略历史交易情况；如果需要获取之前运行策略时产生的历史持仓记录，则需要通过调用该方法来使系统使用本地存储的历史记录，在这种情况下，当因为各种因素（例如在功夫系统外使用别的软件对同一账户手动交易）都会使得功夫内部维护的持仓记录产生偏差，（例如同一账户下对应的不同策略持仓汇总之和不等于账户总持仓），当发生此类偏差时，建议使用缺省模式来从账户持仓恢复策略持仓。

范例

```
# 策略账目所包含的持仓标的列表取决于是否调用 context.hold_book()，但每个标的的具体持仓数值则由 context.hold_positions()来决定，当缺省即没有调用该方法时，策略启动后的账目中的标的持仓等于所对应账户下的标的持仓：
context.subscribe(source, ['600000', '600001'])
# 当调用 hold_positions() 方法后，策略启动后的账目中标的持仓等于上次运行策略结束时所对应的标的持仓：
context.hold_positions()
```

context.book

> 策略的投资组合

类型
book 对象

范例

```
获取策略的投资组合，并打印相关参数
book = context.book
context.log.warning("[strategy capital] (avail){} (margin){}".format(book.avail, book.margin))
```

context.get_account_book(SOURCE, ACCOUNT)

> 账户的投资组合

类型
book 对象

范例

```
# 获取账户的投资组合，并打印相关参数
book = context.get_account_book(SOURCE, ACCOUNT)
context.log.warning("[account capital] (avail){} (margin){}
".format(book.avail, book.margin))
```

辅助函数

context.log.info

> 输出INFO级别 Log 信息

参数	类型	说明
msg	str	Log信息

返回

返回	类型	说明
无	无	无

范例

```
context.log.info(msg)
```

context.log.warning

> 输出WARN级别Log信息

参数

参数	类型	说明
msg	str	Log信息

返回

返回	类型	说明
无	无	无

范例

```
context.log.warning(msg)
```

context.log.error

> 输出ERROR级别Log信息

参数

参数	类型	说明
msg	str	Log信息

返回

返回	类型	说明
无	无	无

范例

```
context.log.error(msg)
```

context.add_timer

> 注册时间回调函数

参数

参数	类型	说明
nano	long	触发回调的纳秒时间戳
callback	object	回调函数

返回

返回	类型	说明
无	无	无

范例

```
# 通过时间回调函数，在1s后撤去订单号为order_id的报单
context.add_timer(context.now() + 1*1000000000, lambda ctx, event:
cancel_order(ctx, order_id))
```

常量定义

Source 柜台

值	说明
"ctp"	CTP柜台
"tora"	华鑫股票柜台
"ToraOption"	华鑫期权柜台
"sim"	模拟柜台

Exchange 交易所

属性	值	说明
SSE	"SSE"	上交所
SZE	"SZE"	深交所
SHFE	"SHFE"	上期所
DCE	"DCE"	大商所
CZCE	"CZCE"	郑商所
CFFEX	"CFFEX"	中金所
INE	"INE"	能源中心

InstrumentType 代码类型

属性	说明
Unknown	未知
Stock	股票
Future	期货
Bond	债券
StockOption	股票期权

Price_Type 报单类型

股票

交易所	交易类型	属性	limit_price
上交所	限价	Limit	非0限价
上交所	最优五档成交剩余转撤销	Any 或者 FakBest5	0
上交所	最优五档成交剩余转限价	ReverseBest	非0限价
上交所（科创板）	本方最优	ForwardBest	0
上交所（科创板）	对手方最优	ReverseBest 或者 Any	0
上交所（科创板）	盘后定价交易	不支持	不支持
深交所	限价	Limit	非0限价
深交所	对手方最优	ReverseBest	非0限价
深交所	本方最优	ForwardBest	非0限价
深交所	立即成交剩余撤销	Fak 或者 Any	0
深交所	最优五档成交剩余撤销	FakBest5	0
深交所	全额成交或撤销	Fok	0

表中覆盖了沪深两所股票(含科创板)普通买卖目前支持的下单类型，其中可转债只支持限价单

客户填写不匹配的参数会导致不可预测的后果，包括但不限于：

- 报单被拒绝
- 报单成功但不是期望的类型

期货

属性	交易类型	limit_price
Limit	限价	非0限价
Any	市价	0
Fak	即时成交剩余撤销	0
Fok	全额成交剩余撤销	0

Side 买卖

属性	说明
Buy	买
Sell	卖
Lock	锁仓
Unlock	解锁
Exec	行权
Drop	放弃行权

Offset 开平

属性	说明
Open	开
Close	平
CloseToday	平今
CloseYesterday	平昨

Direction 多空

属性	说明
Long	多
Short	空

OrderStatus 委托状态

属性	说明
Unknown	未知
Submitted	已提交
Pending	等待
Cancelled	已撤单
Error	错误
Filled	已成交
PartialFilledNotActive	部分成交不在队列中（部成部撤）
PartialFilledActive	部分成交还在队列中

数据结构

Quote 行情信息

属性	类型	说明
source_id	str	柜台ID
trading_day	str	交易日
rcv_time	long	数据接收时间
data_time	long	数据生成时间
instrument_id	str	合约ID
exchange_id	str	交易所
instrument_type	str	合约类型
pre_close_price	float	昨收价
pre_settlement_price	float	昨结价
last_price	float	最新价
volume	int	数量
turnover	float	成交金额
pre_open_interest	float	昨持仓量
open_interest	float	持仓量
open_price	float	今开盘
high_price	float	最高价
low_price	float	最低价
upper_limit_price	float	涨停板价
lower_limit_price	float	跌停板价
close_price	float	收盘价
settlement_price	float	结算价
bid_price	list of float	申买价
ask_price	list of float	申卖价
bid_volume	list of float	申买量
ask_volume	list of float	申卖量

Entrust 逐笔委托

属性	类型	说明
source_id	str	柜台ID
trading_day	str	交易日
data_time	long	数据生成时间
instrument_id	str	合约ID
exchange_id	str	交易所代码
instrument_type	InstrumentType	合约类型
price	float	委托价格
volume	long	委托量
side	Side	委托方向
price_type	PriceType	订单价格类型（市价、限价、本方最优）
main_seq	long	主序号
seq	long	子序号

Transaction 逐笔成交

属性	类型	说明
source_id	str	柜台ID
trading_day	str	交易日
data_time	long	数据生成时间
instrument_id	str	合约ID
exchange_id	str	交易所代码
instrument_type	InstrumentType	合约类型
price	float	成交价
volume	long	成交量
bid_no	long	买方订单号
ask_no	long	卖方订单号
exec_type	ExecType	SZ: 成交标识
bs_flag	BsFlag	SH: 内外盘标识
main_seq	long	主序号
seq	long	子序号

Order 订单回报

属性	类型	说明
rcv_time	long	数据接收时间
order_id	long	订单ID
insert_time	long	订单写入时间
update_time	long	订单更新时间
trading_day	str	交易日
instrument_id	str	合约ID
exchange_id	str	交易所ID
account_id	str	账号ID
client_id	str	Client ID
instrument_type	str	合约类型
limit_price	float	价格
frozen_price	float	冻结价格（市价单冻结价格为0.0）
volume	int	数量
volume_traded	int	成交数量
volume_left	int	剩余数量
tax	float	税
commission	float	手续费
status	str	订单状态
error_id	int	错误ID
error_msg	str	错误信息
parent_id	int	母订单ID
side	str	买卖方向
offset	str	开平方向
price_type	str	价格类型
volume_condition	str	成交量类型
time_condition	str	成交时间类型

Trade 订单成交

属性	类型	说明
rcv_time	long	数据接收时间
order_id	long	订单ID
parent_order_id	long	母订单ID
trade_time	long	成交时间
instrument_id	str	合约ID
exchange_id	str	交易所ID
account_id	str	账号ID
client_id	str	Client ID
instrument_type	str	合约类型
side	str	买卖方向
offset	str	开平方向
price	float	成交价格
volume	int	成交量
tax	float	税
commission	float	手续费

Bar k线行情

属性	类型	说明
trading_day	str	交易日
instrument_id	str	合约代码
exchange_id	str	交易所代码
start_time	int	开始时间
end_time	int	结束时间
open	double	开始价格
close	double	结束价格
low	double	最低价格
high	double	最高价格
volume	int	区间交易量
start_volume	int	初始交易量
tick_count	int	区间有效tick

Book 投资组合

属性	类型	说明
asset	asset	投资组合资金信息
long_position	List of Position	投资组合的持仓列表，对应多头仓位
short_position	List of Position	投资组合的持仓列表，对应空头仓位

获取投资组合持仓列表范例

```
# 通过context.log.info打印组合投资的多头持仓列表
positions = context.get_account_book(SOURCE, ACCOUNT).long_positions
for key in positions:
      pos = positions[key]
      context.log.info("(instrument_id){} (direction){} (volume){}
(yesterday_volume){} ".format(pos.instrument_id,
                          pos.direction, pos.volume, pos.yesterday_volume))
```

Book.asset 投资组合资金信息

属性	类型	说明
avail	float	可用资金
margin	float	保证金
market_value	float	市值
initial_equity	float	初始权益
dynamic_equity	float	动态权益
static_equity	float	静态权益
realized_pnl	float	已实现盈亏
unrealized_pnl	float	未实现盈亏

book.get_position

> 投资组合持仓明细

参数

参数	类型	说明
instrument_id	str	合约ID
exchange_id	Exchange	交易所ID
direction	Direction	持仓方向

返回

返回	类型	说明
position	Position对象	账户持仓明细

范例

```
book = context.get_account_book(SOURCE, ACCOUNT)
context.log.info("[pos]{}".format(book.get_position(
Direction.Long,Exchange.SSE,"600000").volume))
```

Position 持仓信息

期货持仓

属性	类型	说明
instrument_id	str	合约ID
instrument_type	InstrumentType	合约类型
exchange_id	str	交易所
direction	Direction	方向
volume	long	总仓
yesterday_volume	long	左仓
last_price	float	最新价
settlement_price	float	结算价
pre_settlement_price	float	昨结价
avg_open_price	float	开仓均价
position_cost_price	float	持仓成本
margin	float	保证金
position_pnl	float	持仓盈亏
realized_pnl	float	已实现盈亏
unrealized_pnl	float	未实现盈亏

股票持仓

属性	类型	说明
instrument_id	str	合约ID
instrument_type	InstrumentType	合约类型
exchange_id	str	交易所
direction	Direction	方向
volume	long	总仓
yesterday_volume	long	左仓
last_price	float	最新价
pre_close_price	float	昨收价
close_price	float	收盘价
avg_open_price	float	开仓均价
position_cost_price	float	持仓成本
realized_pnl	float	已实现盈亏
unrealized_pnl	float	未实现盈亏

功夫自带python库（pipfile）

```
[source]
url = "http://mirrors.aliyun.com/pypi/simple/"
verify_ssl = false
name = "pypi"

[packages]
conan = "==1.22"
pywin32 = {version = "==227",sys_platform = "== 'win32'"}
pyinstaller = "==3.6"
click = "==7.1.1"
tabulate = "==0.8.6"
prompt_toolkit = "==1.0.14"
PyInquirer = "==1.0.3"
psutil = "==5.7.0"
chinesecalendar = "==1.4.0"
zhdate = "==0.1"
schema = "==0.7.1"
rx = "==3.0.1"
numpy = "==1.16.4"
pandas = "==0.24.2"
statsmodels = "==0.10.1"
```

```
sortedcontainers = "==2.1.0"
recordclass = "==0.12.0.1"
dotted_dict = "==1.1.2"
plotly = "==4.0.0"
tushare = "==1.2.39"

[dev-packages]
clang-format = "==9.0.0"
```