OCTOPUS FOR XTP

交易网关服务器（开发版）

接口说明书

文档编号：OFX02-20180718

一、声明：

OCTOPUS FOR XTP（简称 OFX）是专门为 XTP 后端柜台系统研发的前置接口软件，**目前 OFX 正处于快速迭代开发的状态，因此以下内容随时可能因 OFX 软件升级而不再适用，务请注意！**

二、概要介绍

OFX 是专门为 XTP 后端服务器设置的前置网关软件，提供了通过标准 TCP 协议访问的网络接口，无论使用何种编程语言，只要通过 TCP 协议连接到指定服务器端口，就能通过发送报文指令获取相应数据，执行相应功能。其主要优点有：

1 方便各种编程语言开发。XTP 官方提供的接口文件是 C++风格的动态链接库，在使用其他编程语言时多有不便。OFX 提供标准 TCP 接口开发方式，客户端开发时摆脱了对官方接口库的依赖，更为简单方便。

2 方便实现多客户端连接同一账号。直接使用 XTP 官方 SDK 时，如果要实现多客户端连接同一账号时，需要自行实现并发连接等功能，工作量较大。OFX 已实现单点登陆和指令队列等功能，可直接使用。

3 方便部署。直接使用 XTP 官方 SDK 时，由于 XTP 部署时存在一定合规要求，流程较为复杂，使用网关模式可以减少不少麻烦。OFX 是一整套基于 Linux 操作系统的软件系统，可以方便地进行安装和部署。

4 可方便实现统一存储，扩充旁路功能。XTP 官方 SDK 仅提供较为底层的初级功能，通过 OFX 可以实现统一数据存储，并可方便地扩充旁路功能。

5 高度可定制。OFX 采用报文系统以异步方式进行工作，采用高度模块化架构，并采用松耦合模式，可在不改变前端接口前提下进行局部升级。

三、特性说明

1 一个 XTP 账号对应一个 OFX 交易网关服务器。XTP 账号必须在 OFX 交易网关服务器软件部署时进行绑定，客户端连接 OFX 时使用 OFX 里用户自行设置的用户名和密码登陆。

2 OFX 交易网关服务器采用加密指令上传和数据压缩下发，即所有客户端发出的指令均需进行加密后上传到 OFX 服务器，返回数据则采用经过压缩的 JSON 格式返回。OFX 提供加密算法和解压缩算法组件，密钥在 OFX 交易网关服务器软件部署时进行设置。

3 如客户拥有多个 XTP 账号时，可以采用分布的部署方式，即为每个 XTP 账号分配一个 OFX 交易网关服务器，而将登陆服务部署到另外的服务器上，通过单点登陆时返回的路由信息将客户端路由到不同的交易网关服务器上。

4 网关服务器软件不限制客户端连接数、请求次数、请求频率等，它试图在硬件和网络条件允许前提下尽可能提供所能达到的最高性能。

5 由于 XTP 方面的限制，一个 XTP 账号在同一时间只能保持一个连接，因此当网关服务器软件要连接到 XTP 柜台时，其他使用同一 XTP 账号的软件应断开与 XTP 柜台的连接。

四、网关配置

OFX 交易网关服务器在安装时必须配置好以下参数

1 OFX 密钥：用于报文加密解密的密钥，一般为 8 位以上的字符串，用户可自行设置。

2 XTP 行情服务器地址和端口：由 XTP 提供。

3 XTP 交易服务器地址和端口：由 XTP 提供。

4 XTP 账号、密码和交易密钥：由 XTP 提供。

5 OFX 登陆服务器地址和端口：根据服务器情况设置。

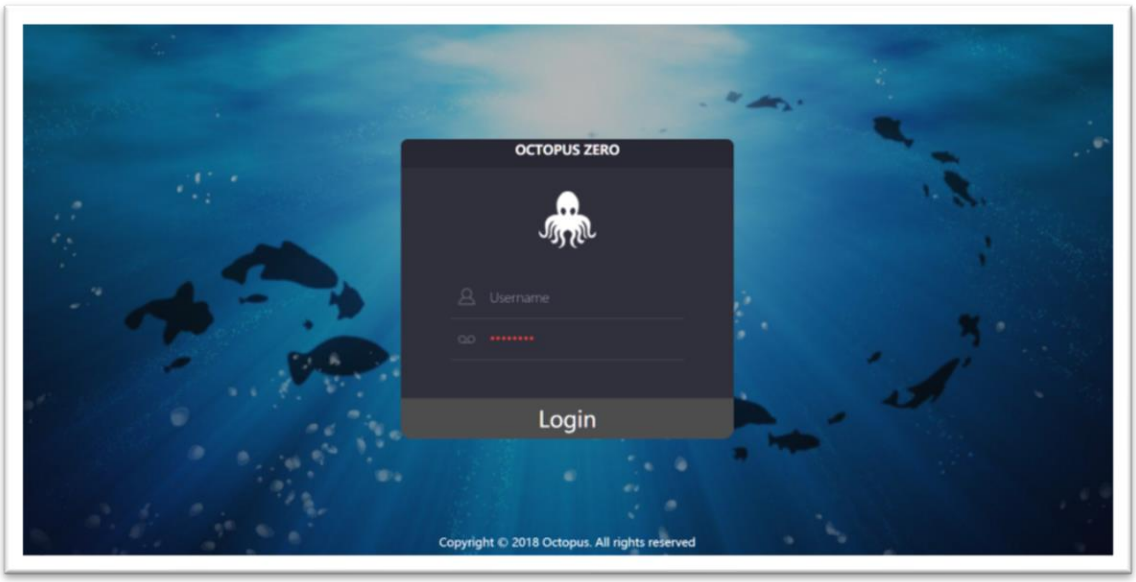
6 OFX 行情服务地址和端口：根据服务器情况设置。

7 OFX 交易服务地址和端口：根据服务器情况设置。

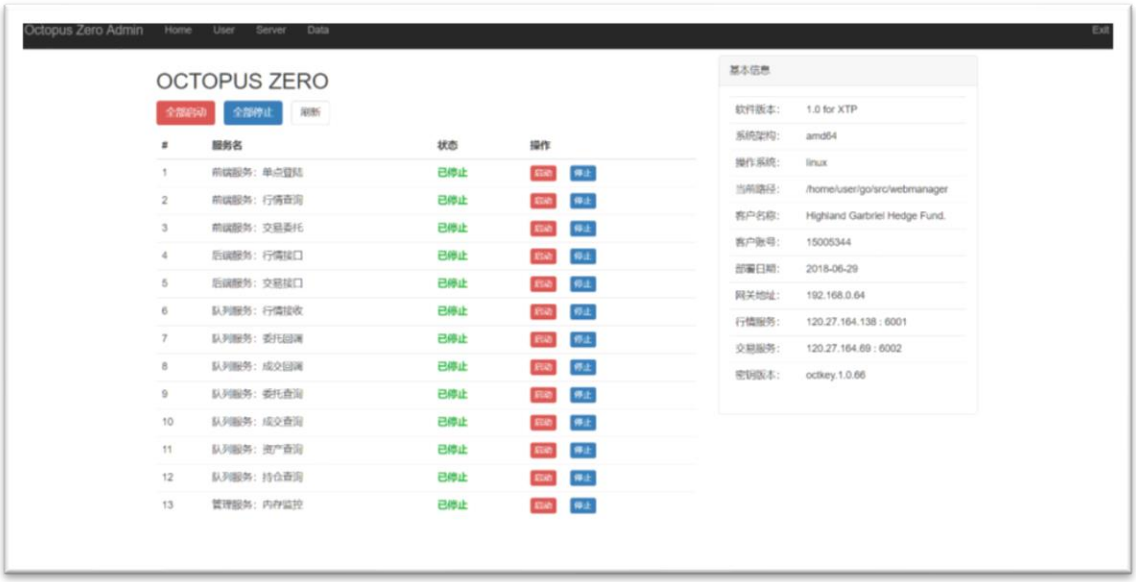
五、网关管理

1 OFX 交易网关服务器软件提供基于 web 的服务器管理工具，可以远程启动或停止各项服务、管理客户端账号信息、查看服务器设置、查询当日数据等。主要功能简介如下：

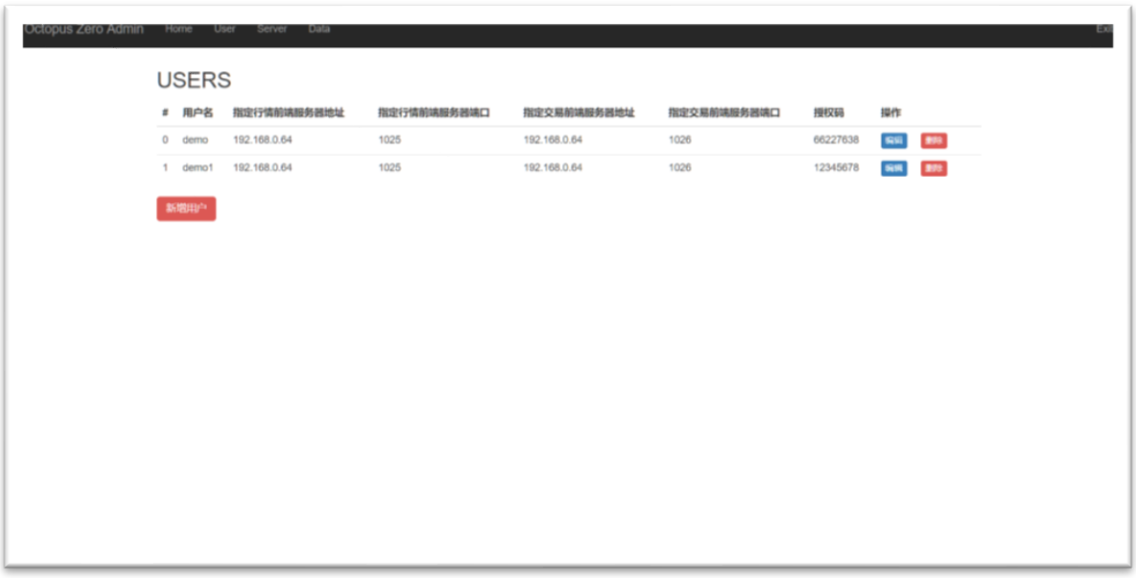
2 登陆：



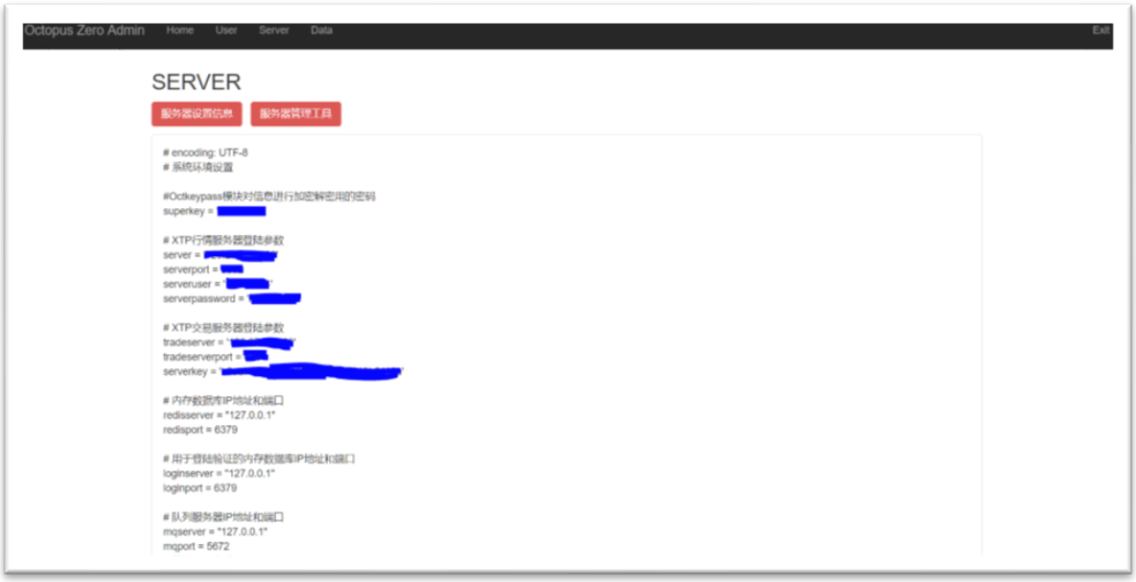
3　启动和停止服务：



4　管理客户端账号：



5 查看服务器设置：



六、报文规则

1 报文命令执行流程：

1） 第一步：组合生成 OFX 报文 JSON；

2） 第二步：将报文 JSON 通过 OCTKEYPASS 加密；

3） 第三步：将加密后字符串发送到指定的登陆服务端口、行情服务端口、交易服务端口；

4） 第四步：读取服务端口返回信息，解压缩后得到 JSON 响应信息；

5） 第五步：解析 JSON 响应信息，做相应后续处理。

2 报文命令执行方式：

报文命令有两种执行形式，一种是同步执行，如登陆报文，服务器会立即返回相应信息；另一种是异步执行，如请求资产报文，服务器收到报文后发送给 XTP 柜台，不会直接返回资产数据，请求到的资产数据需要通过另外的报文命令进行查询。

3 报文指令模板：

1） 报文命令为 JSON 格式，包含以下字段：

（1） Cmd：报文代码；

（2） Authcode：授权码（仅向交易服务端口发送指令时需要）；

（3） Params：JSON 字符串，或者为空。

2） 示例：

（1） 登陆报文的 JSON 字符串表示为：

A { “cmd” : 000001, “params” : {“username” : ”demo”, “password” : ”demo” }}

4 标准状态应答信息

标准状态应答信息为 JSON 字符串，包含以下字段：

1） Stat：200 表示报文已经被网关接受，并转发到 XTP 柜台。

2） Reqid：网关向 XTP 柜台提交指令时使用的请求编号，该编号由网关服务器生成，可用于后续查询数据。

七、报文列表

1 登陆到网关

1） 报文代码：000001。

2） 参数：

（1） Username：客户端登陆网关用户名

（2） Password：客户端登陆网关密码

3） 返回信息：

（1） 包含分配给该用户的网关服务器信息的 JSON 字符串，字段如下：

（2） Quoteserver：行情服务 IP 地址；

（3） Quoteport：行情服务端口；

（4） Tradeserver：交易服务 IP 地址；

（5） Tradeport：交易服务端口；

（6） Authcode：授权连接码，每次发送交易相关报文时需附上本参数。

2 订阅股票（一支或一组）

1） 报文代码：110001。

2） 参数：包含一组股票数据的 JSON 数组，数组每一项都包含 Code（股票代码）和 Market（所属市场）字段。

3） 返回信息：标准状态应答信息，状态代码 200，表示指令已被网关接受并转发到 XTP 柜台。

3 查询单只股票最新行情数据（仅一条）

1） 报文代码：100001。

2） 参数：

（1） Code：股票代码；

（2） Market：所属市场，SH 或 SZ

3） 返回信息：对应 XTP 中 DepthMarketData 数据结构的 JSON 字符串。

4 查询单只股票当日收到的所有行情数据

1） 报文代码：100002。

2） 参数：

（1） Code：股票代码；

（2） Market：所属市场，SH 或 SZ

3） 返回信息：JSON 数组，其中每项都是一个对应 XTP 中 DepthMarketData 数据结构的 JSON 字符串。

5 查询一组股票的最新行情数据

1） 报文代码：100003。

2） 参数：包含一组股票数据的 JSON 数组，数组每一项都包含 Code（股票代码）和 Market（所属市场）字段。

3） 返回信息：JSON 数组，其中每项都是一个对应 XTP 中 DepthMarketData 数据结构的 JSON 字符串。

6 请求资产信息

1） 报文代码：210001。

2） 参数：无。

3） 返回信息：标准状态应答信息，状态代码 200，表示指令已被网关接受并转发到 XTP 柜台。柜台返回的资产数据需另外发送查询报文获取。

7 请求持仓信息

1） 报文代码：210002。

2） 参数：无。

3） 返回信息：标准状态应答信息，状态代码 200，表示指令已被网关接受并转发到 XTP 柜台。柜台返回的持仓数据需另外发送查询报文获取。

8 请求委托信息

1） 报文代码：210003。

2） 参数：无。

3） 返回信息：标准状态应答信息，状态代码 200，表示指令已被网关接受并转发到 XTP 柜台。柜台返回的委托数据需另外发送报文获取。

9 请求成交信息

1） 报文代码：210004。

2） 参数：无。

3） 返回信息：标准状态应答信息，状态代码 200，表示指令已被网关接受并转发到 XTP 柜台。柜台返回的成交数据需另外发送报文获取。

10 查询委托事件

1） 报文代码：200001。

2） 参数：包含 Code 和 Market 字段的 JSON 字符串。

3） 返回信息：多条对应 XTP 中 XTPOrderInfo 结构的 JSON 数组字符串。

11 查询成交事件

1） 报文代码：200002。

2） 参数：包含 Code 和 Market 字段的 JSON 字符串

3） 返回信息：多条对应 XTP 中 XTPTradeReport 结构的 JSON 数组字符串。

12 查询资产信息

1） 报文代码：200003。

2） 参数：无。

3） 返回信息：一条对应 XTP 中 XTPQueryAssetRsp 结构的 JSON 字符串。

13 查询持仓信息

1） 报文代码：200004。

2） 参数：Reqid（可选）或留空。

3） 返回信息：如指定了 Reqid 参数，则返回对应 Reqid 的信息，否则返回包含历史请求的全部柜台返回信息。返回多条对应 XTP 中 XTPQueryTraderReq 结构的 JSON 数组字符串。

14 查询委托信息

1） 报文代码：200005。

2） 参数：Reqid（可选）或留空。

3） 返回信息：如指定了 Reqid 参数，则返回对应 Reqid 的信息，否则返回包含历史请求的全部柜台返回信息。返回多条对应 XTP 中 XTPQueryOrderRsp 结构的 JSON 数组字符串。

15 查询成交信息

1） 报文代码：200006。

2） 参数：Reqid（可选）或留空。

3） 返回信息：如指定了 Reqid 参数，则返回对应 Reqid 的信息，否则返回包含历史请求的全部柜台返回信息。返回多条对应 XTP 中 XTPQueryTradeRsp 结构的 JSON 数组字符串。

16 提交委托（下单）

1） 报文代码：210005。

2） 参数：对应 XTP 中 XTPOrderInsertInfo 结构的 JSON 字符串。

3） 返回信息：标准状态应答信息，状态代码 200，表示指令已被网关接受并转发到 XTP 柜台。委托状态需使用另外报文请求和查询。

17 撤销委托（撤单）

1） 报文代码：210006。

2） 参数：order_xtp_id，XTP 订单号。

3） 返回信息：标准状态应答信息，状态代码 200，表示指令已被网关接受并转发到 XTP 柜台。委托状态需使用另外报文请求和查询。

18 请求 ETF 清单信息

1） 报文代码：210007

2） 参数：market 所属市场+code 股票代码（可选）或空，如参数为空则返回所有 ETF 合约。

3） 返回信息：标准状态应答信息，状态代码 200，表示指令已被网关接受并转发

到 XTP 柜台。ETF 清单信息需使用另外报文请求和查询。

19

八、OCTKEYPASS 加密库

1 OCTKEYPASS 加密库是 OFX 软件专门用于报文加密和数据压缩的运行库，包含四个标准函数，分别用于加密、解密、压缩、解压缩，目前有 Python for Windows（Pyd）、Windows DLL（C Style）、Linux SO 三种格式。

2 加密解密时用到的密钥由用户自行设置，需与网关服务器设置密钥保持一致。

3 用户可自行开发加密库，用来替代 OFX 自带的库文件。

九、示例程序

以下演示程序演示了使用 Python 语言连接到登陆服务端口，组合报文、加密报文并提交、获取返回信息的过程，示例程序中的地址和端口等参数请自行修改，其他编程语言可参照开发。

```
# encoding: UTF-8

from mods import octkeypass
from socket import *
import json
import config
import time

host = "218.57.111.30"
port = 11027
buffersize=1
addr = (host,port)

client = socket()
client.connect(addr)

print 'connected'

def test_000001():
    cmd = dict()
    cmd['cmd']=000001
    p = dict()
    p['username']='demo'
    p['password']='demo'
    cmd['params']=p
    cstr = json.dumps(cmd)
    #print len(cstr),config.superkey, type(config.superkey)
    cstr = octkeypass.encrypt(config.superkey, cstr)
    #print cstr
    client.sendall("%s\r\n" % cstr)
    c= client.makefile().readline()
    #print c
    print octkeypass.extract(c)

test_000001()

client.close()
```

运行结果：

```
user@user-VirtualBox:~/server$ python login_front_test.py
connected
{"tradeserver": "218.57.111.30", "authcode": "66227638", "tradeport": 11026, "un
ame": "demo", "quoteserver": "218.57.111.30", "quoteport": 11025}
user@user-VirtualBox:~/server$
```

1