普兰泰科报价引擎客户端使用说明书

V1.0

目 录

[客户使用接口类图 3](#_Toc395887119)

[1 IClientConneted 3](#_Toc395887120)

[2 IBussiness 4](#_Toc395887121)

1. **客户使用接口介绍**

普兰泰科为专业报价行情提供访问接口并负责维护该接口.如果该接口有更新将第一时间通知客户.仅限于普兰泰科科技有限公司和客户交流资料.

1. **普兰泰科报价引擎接口特点和优势.**

* 接口数量少,现总共只有两个接口实现了所有用户需要的功能.
* 接口的方法简单明了,参数名称已表达了参数意义,功能定义一目了然.
* 行情数据载体采用java的pojo对象和一个字符串数组.最简化的载体方便操作且不失可扩展性.用户可用jar包中的工具类将java POJO对象转换成JSON,XML或者纯String格式.方便用户使用行情数据.

# 接口类图





# IClientConneted接口

**接口功能说明** : 使用该接口的方法向报价引擎发起登录请求和订阅请求两个功能.

实现类:ClientConnector

* + 1. **接口方法**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **方法名称** | **返回值** | **方法说明** | **参数** | **参数说明** |
| ClientConnector(bussinessHandler) |  | 接口实现类构造方法. | bussinessHandler: IBussiness | 数据接收处理接口的实现类作为参数. |
| conneteLocateGateWay | void | 向报价引擎登录注册客户信息.参数由普兰泰科提供.可以将参数配置到本地文件,方便随时修改. | serverAddress: String  port: int  userName:String  password: String | 报价引擎地址  报价引擎端口  用户名  用户密码 |
| openRICMarket | void | 向报价引擎发起订阅产品请求.RIC编码规则由普兰泰科提供. | RIC: String | RIC订阅产品编码 |

* 1. **示例程序**

//接口声明并用实现类实例化.

IClientConneted clientConnetor = new ClientConnector(bussinessHandler);

//向服务器登录并注册客户端信息.参数1:服务器ip,参数2:服务器端口,参数3:客户名称,参数4:客户密码.

sample.clientConnetor.conneteLocateGateWay("61.144.244.173", 8888, "ztcj", "ztcj2013");

//向服务器发送某个产品订阅请求.

//订阅现货金

sample.clientConnetor.openRICMarket("XAU=");

//订阅现货银

sample.clientConnetor.openRICMarket("XAG=");

//用户可以订阅多支产品.

1. **IBussiness接口**
   1. **接口说明**

报价引擎调用该接口的的方法进行行情数据处理, 服务端异常处理

注:客户自行实现该接口中的抽象方法进行行情数据的处理.Jar档中没有实现该接口的的类供客户调用,只提供了一个RFASample程序给客户参考.

* 1. **接口方法**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **方法名称** | **返回值** | **方法说明** | **参数** | **参数说明** |
| handleMessage | void | 处理报价引擎发送过来的报价信息. | message: LocateUnionMessage | 报价引擎发送的报价,格式一般java POJO对象LocateUnionMessage |
| handleException | void | 处理通讯,系统异常 | e: Throwable | 异常具体信息 |
| handleDisconnected | Void | 处理网络连接异常 | 无 | 无 |

* 1. 示例

package com.locate.sample;

import java.util.List;

import javax.xml.bind.JAXBContext;

import javax.xml.bind.JAXBException;

import javax.xml.bind.Marshaller;

import net.sf.json.JSONObject;

import org.apache.log4j.Logger;

import org.dom4j.io.DocumentResult;

import com.locate.common.LocateMessageTypes;

import com.locate.common.model.LocateUnionMessage;

import com.locate.common.utils.NetTimeUtil;

import com.locate.face.IBussiness;

import com.locate.face.IClientConnector;

import com.locate.gate.handler.ClientConnector;

/\*\*

\* 客户端接口调用示例

\* @author CloudWei kaiweicai@163.com

\* create time 2014年8月26日

\* @copyRight by Author

\*/

public class RFASample {

private Logger logger = Logger.getLogger(getClass());

public static final String XAU\_RIC = "XAU=";

public static final String XAG\_RIC = "XAG=";

//客户端接口声明.

public IClientConnector clientConnetor;

public IBussiness bussinessHandler;

//XML对象处理器上下文.

private JAXBContext context;

//实例化

public RFASample(){

//客户端收到数据后调用该接口的方法进行数据处理.因为每个客户数据处理的需求不同,所以客户需要实现该接口进行具体处理数据业务.

bussinessHandler = new BussinessHandler();

//向服务器发送请求的接口.这个接口客户无需实现.只要调用这个ClientConnector这个类里面的方法就好了.

clientConnetor = new ClientConnector(bussinessHandler);

//数据格式转换前置工作.客户可以不使用.

try {

context = JAXBContext.newInstance(LocateUnionMessage.class);

} catch (JAXBException e) {

logger.error("Json initial JAXBContext error!",e);

}

}

class BussinessHandler implements IBussiness{

/\*

\* 客户端与服务器交互发生异常时的处理方法

\*/

@Override

public void handleException(Throwable e){

//简单的打印处理,可以显示,通知,邮件告警该异常,并联系普兰泰科技术支持

System.out.println(e);

}

/\*

\* 客户需要实现数据处理的代码.

\* 参数message是服务器发送过来统一格式的数据.

\* 客户可以根据需要将message转成XML或者JSON格式.

\*/

@Override

public void handleMessage(LocateUnionMessage message){

long starTime = message.getStartTime();

long endTime = NetTimeUtil.getCurrentNetTime();

//计算从服务器到收到该消息总共耗费的时间,由于使用的是网络时间,所以存在一定的误差.

System.out.println("Recieve this message use time:"+(endTime-starTime)+" microseconds");

byte msgType = message.getMsgType();

String RIC = message.getRic();

switch(msgType){

//首先服务器会发送过来一个行情的全数据信息.里面包括该种类RIC对应的所有字段(报价商名称,昨日收盘价, 昨日开盘价).

case LocateMessageTypes.REFRESH\_RESP:

//如果订阅的是多个产品,此处根据RIC对产品进行分类处理.

switch(RIC){

case XAU\_RIC:

System.out.println(RIC);

System.out.println("handel the au message "+message);

break;

case XAG\_RIC:

System.out.println(RIC);

System.out.println("handel the ag message "+message);

break;

}

break;

//然后服务器会发送不断更新的MarketPrice.该信息只包括需要更新的字段.

//如RIC: XAU= 只会发送BID,BID1,BID2,ASK,ASK1,ASK2等行情变化的字段.

case LocateMessageTypes.UPDATE\_RESP:

//如果订阅的是多个产品,此处根据RIC对产品进行分类处理.

switch(RIC){

case XAU\_RIC:

System.out.println(RIC);

System.out.println("handel the au message "+message);

//具体解析一个报价出来看看是什么样子的.

List<String[]> payloadList = message.getPayLoadSet();

for(String[] payLoad:payloadList){

System.out.println("payload id is:"+payLoad[0]);

System.out.println("payload name is:"+payLoad[1]);

System.out.println("payload type is:"+payLoad[2]);

System.out.println("payload value is:"+payLoad[3]);

}

break;

case XAG\_RIC:

//将消息对象转换成JSON型字符串使用.

JSONObject jsonObject = JSONObject.fromObject(message);

String jsonResult = jsonObject.toString();

System.out.println(RIC);

System.out.println("handel the XAG JSON format!"+jsonResult);

try {

// 下面代码演示将对象转变为xml

Marshaller marshaller = context.createMarshaller();

DocumentResult node = new DocumentResult();

marshaller.marshal(message, node);

System.out.println("handel the XML format!"+node.getDocument().asXML().toString());

} catch (JAXBException e) {

e.printStackTrace();

}

}

break;

//向服务器发送请求后的返回信息.

case LocateMessageTypes.SERVER\_STATE:

String errorDescription = message.getResultDes();

System.out.println("服务器处理了请求并返回了服务器的处理结果.");

String state = message.getState();

System.out.println(state);

System.out.println(errorDescription);

break;

//如果服务器有通知服务器状态改变的信息,会使用此状态信息.

case LocateMessageTypes.STATUS\_RESP:

errorDescription = message.getResultDes();

System.out.println("服务器出现了新的状态.");

state = message.getState();

System.out.println(state);

System.out.println(errorDescription);

break;

//服务器发送了未知的消息,一般这里不用处理.扔掉该消息就好了.或者打印一个警告.

default:

System.out.println("Not should to here! message type is "+LocateMessageTypes.REFRESH\_RESP);

}

// String content = response.asXML();

// System.out.println("Received server's message : " + message+"\n");

}

/\*\*

\* 与服务器的网络连接断开后处理方法

\*/

public void handleDisconnected() {

System.out.println("Locate Server disconnted!!! ");

}

}

public static void main(String[] args) {

RFASample sample = new RFASample();

//向服务器注册客户端信息.参数1服务器ip,参数2服务器端口,参数3客户名称,参数4客户密码.

// sample.clientConnetor.conneteLocateGateWay("61.144.244.173", 8888, "ztcj", "ztcj2013");

sample.clientConnetor.conneteLocateGateWay("127.0.0.1", 8888, "ztcj", "ztcj2013");

//向服务器发送RIC请求.BussinessHandler的handleMessage方法就可以接收到服务器返回的市场价格了.

sample.clientConnetor.openRICMarket(XAU\_RIC);

sample.clientConnetor.openRICMarket(XAG\_RIC);

}

}

1. **普兰泰科科技有限公司简介:**

**普兰泰科科技有限公司坐落于中国第一个特区,深圳市华强北群星广场33楼.**

**是一家新兴的专业致力于国内金融产品报价的资讯公司.我们竭诚为客户提供稳定,及时,准确的金融产品报价,并可根据客户需求定制客户产品报价.与国际接轨是我们的一大特色.**

**普兰泰科对于接口的态度是全力支持客户的业务需求,并及时提供关于接口问题的咨询,问题解答,客户接口具体应用程序实现等.如果您有任何问题请与我们及时联系.**

**普兰泰科科技有限公司通讯地址:**

**普兰泰科科技有限公司技术支持联系电话:**