会员端 IMIX 基础组件
用户使用手册

中国外汇交易中心暨全国银行间同业拆借中心

2019 年 10 月

目 录

1.关于 IMIX 介绍

IMIX 协议全称银行间市场信息交换协议，用于银行间本币市场和外汇市场的金融信息的传输。

IMIX 基础组件是使用 IMIX 协议的消息中间件，它被使用于连接不同的客户端和服务端。用户可以使用 IMIX 基础组件开发自己的客户端，来链接任何使用 IMIX 协议标准的服务端程序。

IMIX 基础组件 API 提供了一系列方法来想对使用 IMIX 协议的消息进行解码，这将使得用户在开发自己的应用时节省很多时间和精力。

2.IMIXClientAPI

IMIXClientAPI 是针对 IMIX 消息传输需求进行开发的一套借口，目的是帮会员端快速接入中心内部系统获取数据，会员端系统只需要简单的调用 API 即可实现与中心内部系统的链接。

2.1 IMIXClientAPI 流程综述

用户必需首先初始化 IMIXApplication，传入配置文件和自己的回调函数。再调用 IMIXSession 的 start 方法来完成登陆。接下来 IMIXSession 会根据用户传入的用户名、口令向服务器方发送 logon 请求并更具响应来完成登陆。当 IMIXSession 收到 logon 返回的消息后，登陆正式完成。如果登陆失败，也将收到对应的错误信息。如图 2 所示

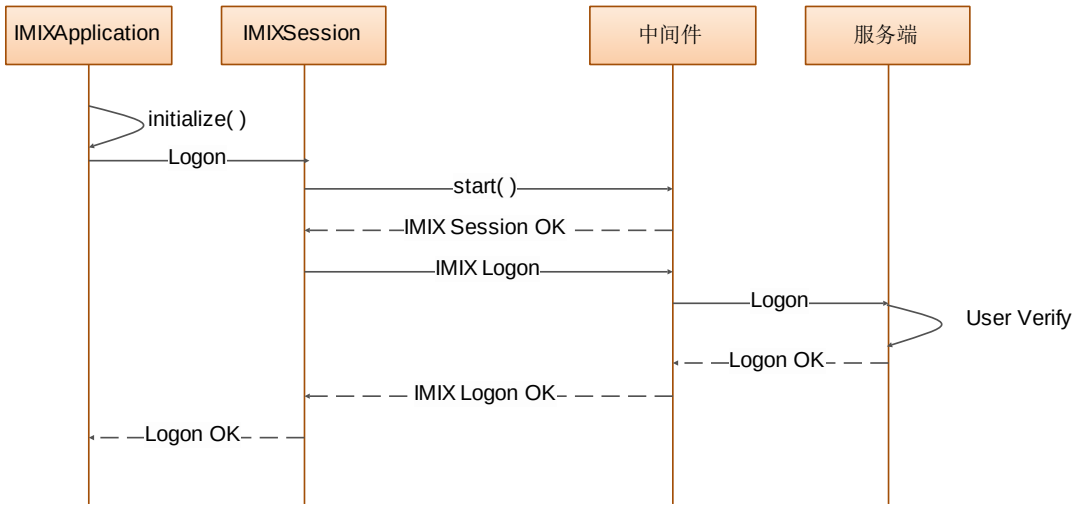


图 1 登录流程

2.1.2 登出

Logout 时，用户可调用 IMIXSession 的 stop 方法或者直接调用 IMIXApplication 的 stop 方法来终止整个程序。如图 3 所示：

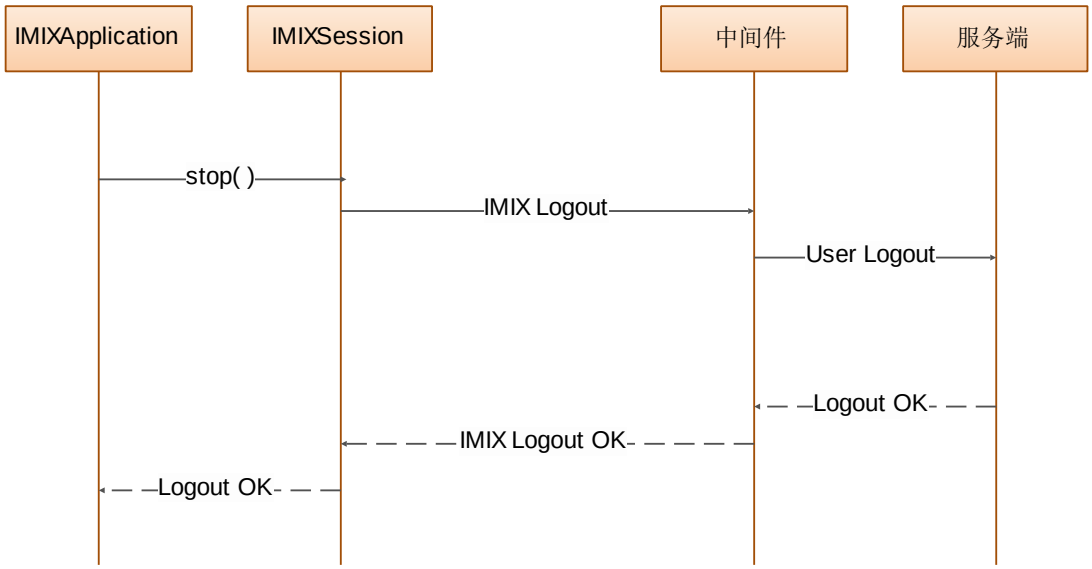


图 2 登出流程

2.1.3 消息交换

用户可以处理两种 IMIX 消息，一种是 Application 消息，一种是 Admin 消息。对开发者而言，只需要发送和通过回调函数接收消息，需要做的是实现自己的回调并且在初始化的时候注册进 IMIXApplicaiton。如图 4 所示：

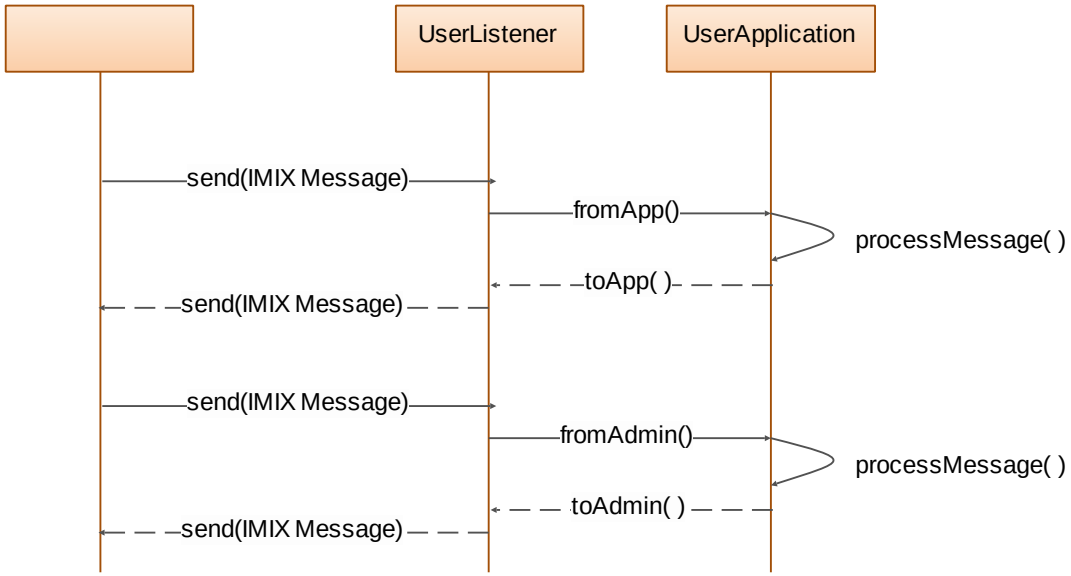


图 3 消息交换流程

2.1.4 事件和异常处理

消息交互，事件和异常处理也是通过回调函数来完成的。如图 5 所示：

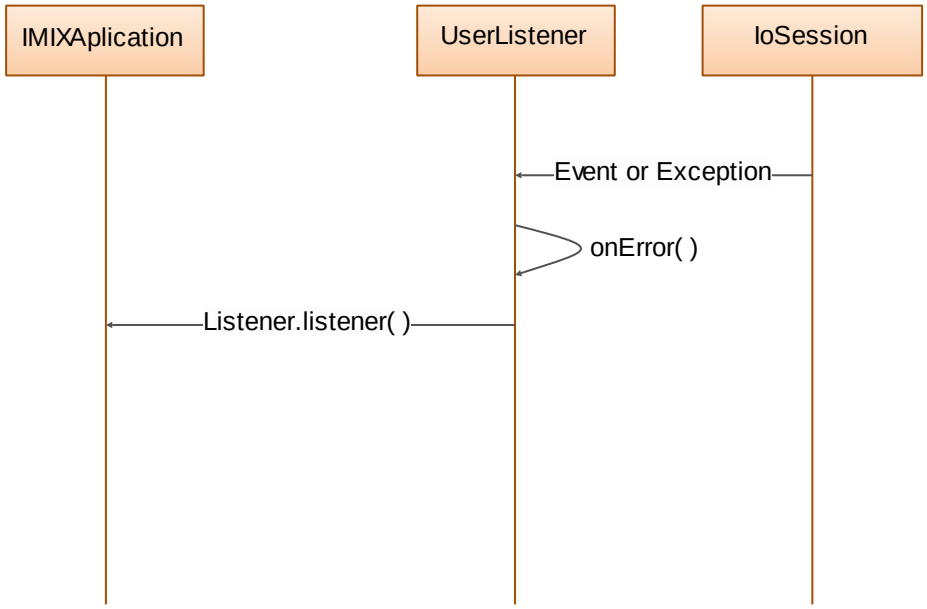


图 4 异常处理流程

2.2.1 IMIXApplicaiton

2.2.1.1 描述

IMIXApplication 是所有的 IMIXSession 的管理架构，用户可拥有多个 IMIXSession 的实例，但是 IMIXApplication 只能有一个。

2.2.1.2 方法

public static void initialize(Listener listener, String configFile)
使用用户自定义的回调和配置文件来初始化 IMIXApplicaiton

public static voidstop()
关闭 IMIXApplication 中所有的 IMIXSession

2.2.2 IMIXSession

2.2.2.1 描述

IMIXSession 代表了连接到 IMIX 中间件上的连接。当 SessionCreate 事件被收到后，IMIXSession 就被创建了。

2.2.2.2 方法

public ImixSession(String username, String password)
根据用户名和密码来构造一个 ImixSession 实例， 其中用户名不能为空或是“”*

public ImixSession(String username, String password, String market)
根据用户名密码和市场来构造一个 ImixSession 实例， 其中用户名不能为空或是“”， 市场信息在构造 ImixSession 的时候不是必须的*

public ImixSession(String username, String password, String senderCompID, String targetCompID, String targetSubID)
根据用户名、密码、发送方 ID、目标方 ID 和目标方子 ID 来构造一个 ImixSession 实例， 其中用户名不能为空或是“”， 市场信息在构造 ImixSession 的时候不是必须的*

public DataDictionary getDataDictionary()
通过 ImixSession 得到字典实例

public boolean isStarted()

检查 IMIXSession 是否已经成功启动

public static ImixSession lookupIMIXSession(Messagemessage)

根据消息找到 IMIXSession

public boolean send(Message message)

向 IMIXSession 发送消息。如果由于某些原因，该消息没有发送成功，用户想再次发送的话，要将消息里的 PossResend 标识设为 true

public synchronized boolean start()

启动 ImixSessioin 并连接到 IMIX 中间件

public synchronized boolean start(boolean resetOnLogon)

启动 ImixSessioin 并连接到 IMIX 中间件.用户可以通过函数的参数来配置是否要在 logon 的时候 reset file stor
e,并且这个参数将会覆盖配置文件中的 ResetOnLogon 的值

public synchronized boolean start(int overTime)

启动 ImixSession 并连接到 IMIX 中间件， 如果在 overTime 时间内仍没有连接成功，将返回 false

public synchronized boolean start(int overTime, boolean resetOnLogon)

启动 ImixSession 并连接到 IMIX 中间件，该函数可以指定是否要在 logon 的时候 reset message store,并且可以指定 overTime 的时间

public synchronized boolean start(String newPassword)

启动 ImixSession 并连接到 IMIX 中间件，该函数可以用来在 logon 的时候修改密码，参数 newPassword 就是 client 要修改成的新密码

public synchronized boolean start(int overTime, boolean resetOnLogon, String newPassword)

启动 ImixSession 并连接到 IMIX 中间件，该函数可以用来设置连接的超时时间，是否在 logon 的时候 reset file store 以及重设密码

```
public void stop()
```

停止并且关闭 IMIXSession.

```
public synchronized void stop(boolean force)
```

允许用户强制关闭 IMIXSession,只要参数传入 true 即可

```
public string getLogonError ()
```

如果返回登出，可以得到登出的错误码，请根据下面错误码对应信息查看错误信息

用户在登录时候，如果返回的消息是 35=A 消息，则代表登录成功；如果返回的消息是 35=5 的消息，则代表登录失败，58 号域的内容就是错误码，再根据下边的错误码来查看对应的错误信息。
注：可以用 getLogonError 方法来取错误码。

2.2.3 Listener (Client)

2.2.3.1 描述

Listener 是一个抽象的 interface，API 的用户必须实现这个接口来管理来自 IMIX 中间件各种事件、异常和消息。

2.2.3.2 方法

```
public void fromApp(Message message, IMIXSession imixSession)
```

当有 App 消息时调用的回调

```
public void fromAdmin(Message message, IMIXSession imixSession)
```

当有 Admin 消息时调用的回调

```
public void onLogon(IMIXSession imixSession)
```

当有 Logon 消息时调用的回调

```
public void onLogout(IMIXSession imixSession)
```

当有 Logout 消息时调用的回调

```
public void toApp(Message message,IMIXSession imixSession)
```

当用户发送 App 消息时，调用的回调

```
public void onError(ImixSession imixSession, int type)
```

当底层发生错误时的回调

2.4 IMIX API 配置

2.4.1 IMIX API 配置

API 用户必须使用配置文件来配置 API，配置的相关大类有：

(1) 远端 Agent 的地址和端口。

(2) 远端市场对应的 Agent 信息，以前的配置文件中 Market 信息是必须的，新版本进行了修改，用户可以不再配置 Market 的信息，API 将通过 UserName 和 SenderSubID 的匹配来查找 ImixSession 的信息。

(3) API 模块中派发消息、异常和事件的处理模型（单线程和多线程等）

(4) 存储消息的工厂类（文件存储、数据库存储等）

(5) 现在的 ImixClientAPI 支持动态的 Session，并且 ImixSession 的用户名和 SenderSubID 是相同的，因此如果用户动态输入的用户名能够在配置文件中找到匹配 SenderSubID，将创建一个静态的 ImixSession,如果找不到匹配的，但是配置文件中有 SenderSubID=*的 TradeSession，将会创建一个动态的 ImixSession。

(6) 新版的 CAEnable 是登录时候用来验证 api 的数字证书，是在登录时握手用的。

(7) 新版的配置文件中的加入了数字证书验证，而数字证书需要的文件有，在配置中 UserCert 和 UserKey 还有 UserCertPwd 的路径需要配置，因为在握手验证的时候是要用到这些文件。

(8) 新版的配置文件中还加入了 ResetOnLogon，在每次重新登录时，是否重置序列号，默认为 Y。

(9) 现在的 ImixClientAPI 支持动态的 Session，并且 ImixSession 的用户名和 SenderSubID 是相同的，因此如果用户动态输入的用户名能够在配置文件中找到匹配 SenderSubID，将创建一个静态的 ImixSession,如果找不到匹配的，但是配置文件中有 SenderSubID=*的 TradeSession，将会创建一个动态的 ImixSession。

(10) 现在的 api 在每次发消息时候要配置好 TargetSunID，填好数字证书相关信息和用户。 这样才能确保连上相对应的服务器。

(11) 在 clientAPI 登录成功后从 FTP 中下载相关文件，所以现在需要配置 FTP 的相关信息以及下载的内容的存放路径。FTP 的配置相关信息在 ftpConfig 文件夹下，其中 cfetsftpclient.jks 是相关证书，用户不用管理，ftpclient.properties 是用户连接目标服务器的配置(其中 activeHost 是目标服务器的 ip,activePort 是目标服务器的端口,userName 是连接目标服务器用户，passWord 是用户密码)。会员可以自由选择在 ftpclient.properties 中配置 FTP 下载文件保存路径(

现券市场:

现券 NDM 报价:ndm_bond_file_save_path

现券 ODM 报价:odm_bond_file_save_path

现券 QDM 报价(请求+指示性):qdm_bond_file_save_path

现券 QDM-ESP 报价:qdm_esp_bond_file_save_path

质押式回购市场:

质押式回购 NDM 报价:ndm_file_save_path

质押式回购 ODM 报价:odm_crpo_save_path

质押式回购 QDM 报价:qdm_crpo_save_path

衍生品市场:

ODM 公用(交易):odm_der_base_save_path

ODM 利率互换(交易):odm_der_irs_save_path

ODM 标准利率互换(交易):odm_der_sirs_save_path

ODM 标准债券远期(交易):odm_der_sbfwd_save_path

ODM 利率互换(行情):odm_md_irs_base_save_path

ODM 标准利率互换(行情):odm_md_sirs_base_save_path

NDM 利率互换(交易):ndm_irs_save_path

说明：配置的存放路径不能带有中文)，log_file__path 日志信息的路径。FTP 下载文件会默认存储到会员本地文件目录，相关目录和数据对应关系如下：

校验报表文件说明	文件相对路径
现券市场 NDM(交易)	data/ndmBondDownload
现券市场 ODM(交易)	data/odmBondDownload
现券市场 QDM(交易)	data/qdmBondDownload
现券市场 QDM-ESP(交易)	data/ qdmEspBondDownload
质押市场NDM(交易)	data/ndmDownload
质押市场ODM(交易)	data/odmCrDownload
质押市场QDM(交易)	data/qdmCrDownload
衍生品市场_公用(交易)	data/odmDownLoad/base
衍生品市场_利率互换(交易)	data/odmDownLoad/irs
衍生品市场_标准利率互换(交易)	data/odmDownLoad/sirs
衍生品市场_标准债券远期(交易)	data/odmDownLoad/sbfwd
衍生品市场_利率互换(行情)	data/odmMdDownload/irs
衍生品市场_标准利率互换(行情)	data/odmMdDownload/sirs
利率互换_对话报价(交易)	data/ndmIrsDownload

（12）client.cfg 文件中配置[default]节点下 RelatedToFTPPath 配置项为跟 ftp 配置相关的路径，默认为项目的根目录下 ftpConfig/ftpclient.properties

（13）调用 clientAPI 中 send（）方法进行消息发送，若校验不通过，日志会打印相

关堆栈信息，同时校验包提供一个接口方法 FirstCheckRules.ReasonByClordID(String clordID),该方法的参数为客户端参考编号，返回值为 Exception 对象，上限为 100 条。

（14）用户登录新本币不同 Session 时，会自动下载不同市场需要的规则校验数据文件。登录的 Session 为现券市场（交易）/质押式回购市场(交易)/衍生品市场(交易、行情)时，Session 登录时，会下载当天最新数据文件到对应文件目录中，现券市场和质押式回购市场根据 session 登录的时间,每 15~17 分钟下载一次最新规则校验数据文件,衍生品市场根据 session 登录的时间,每 60~62 分钟下载一次最新规则校验数据文件。当登录 Session 为现券市场时，部分数据(XML)下载时间点每天 5:58、12:28、15:28、20:58 四个时间点下载对应时间点数据保证数据更新。

（15)规则校验包基于下载到本地的数据文件进行校验，当下载 RDI 数据文件出现问题时（RDI 环境异常、RDI 权限异常等问题），多次下载失败后，会以本地规则校验数据文件进行校验。

注：数字证书需要安装和需要的配置请看附录。

配置项	配置说明
[default]	Session 的默认配置
FileStorePath=data/store	存放 FileStore 文件的路径
QueueFilePath=data/queue	存放消息队列的文件
RollingDays=5	保留文件的日期，超过这天数就会被删除
SLF4JLogPropertiesFilePath=cfg/log4j.properties	Log 配置文件所在的路径
TimeZone=GMT+08:00	指定时区为北京时区
StartDay=Monday	Session 开始的日期
EndDay=Sunday	Session 结束的日期
StartTime=00:00:00	Session 开始的时间
EndTime=23:59:59	Session 结束的时间
ImixReceiveQueueSize=100	接收队列大小(流量控制用)
ImixMaximumConnectionBufferSize=4096	TCP 连接的缓存大小
SLF4JLogEvents=Y	日志中是否记录事件
SLF4JLogIncomingMsgs=Y	日志中是否记录收到的消息
SLF4JLogOutgoingMsgs=Y	日志中是否记录发送的消息

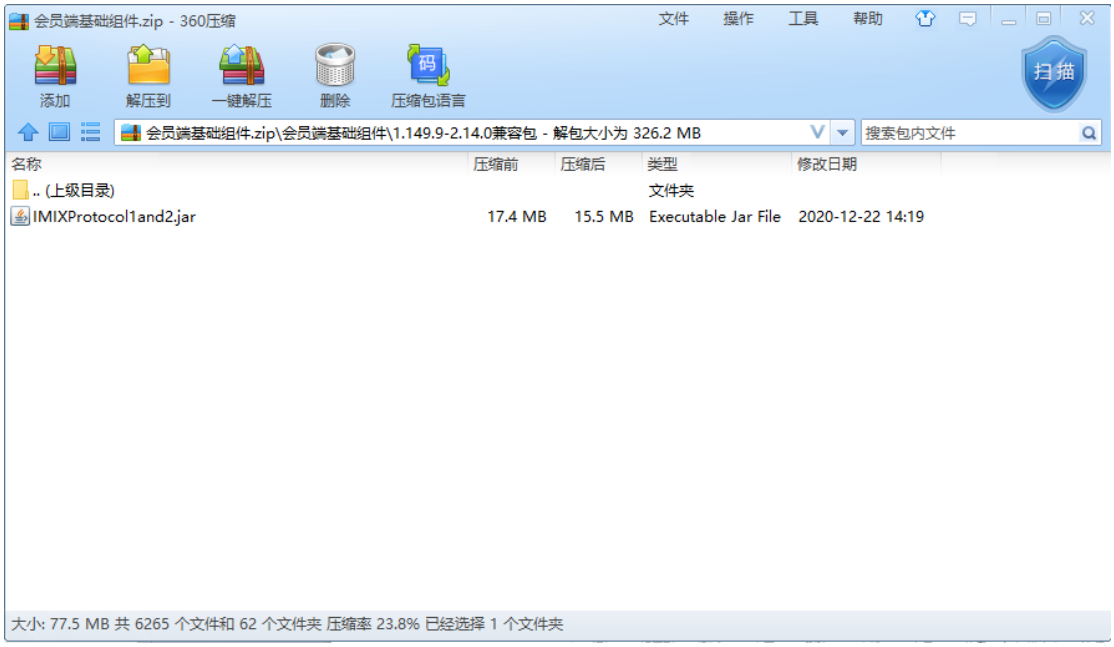
SLF4JLogHeartbeats=Y	日志中是否记录心跳
CAEnable=Y	是否使用 ca
UserCert=cert/UserCert.jks	UserCert 的文件位置，需要握手和初始化登录的时候用到。
UserKey=cert/UserKey.key	UserKey 的文件位置，需要握手和初始化登录的时候用到。
CertChain=cert/CertChain.spc	CertChain 的文件位置，需要握手和初始化登录的时候用到。
UserCertPwd=e2e55b53	数字证书的密码
ResetOnLogon=Y	每次登陆是否重置序列号
[session]	Session 定义
BeginString=IMIX.1.0	协议版本号
ConnectionType=initiator	连接的类型(initiator 表示发起方,Acceptor 表示接收方)
SenderCompID=haiquan00000000000000	Client 在这个 Session 中本身的名称,机构代码(请咨询交易中心获取对应的结构代码)
ServiceID=CFETS-TRADING-INFT	连接的服务名称
TargetSubID=CSWAP/XSWAP	发给那个服务器
SenderSubID=haiquan	Client 在这个 Session 中本身的子名称(*表示动态输入)
SocketUseSSL=Y	是否使用 ssl
SocketConnectHost=200.31.145.211	目标 Agent 的 IP(或者 ExternalHub 的 IP)
SocketConnectPort=9883	目标 Agent 的 Port（或者 ExternalHub 的 Port）
ReconnectInterval=5	睡眠时间，在链接失败或者重新发送之后
LogonTimeout=90	登陆超时的时间(单位为秒)
HeartBtInt=90	发送心跳消息的间隔
AutoConnect=N	Session 被远端断了以后是否自动重连
CheckLatency=N	是否校验时间

[monitor]	监控服务器配置
MonitorServerIp=127.0.0.1	监控服务器地址
MonitorServerPort=8997	监控服务器端口

client.cfg

2.4.2 开发包替换详解

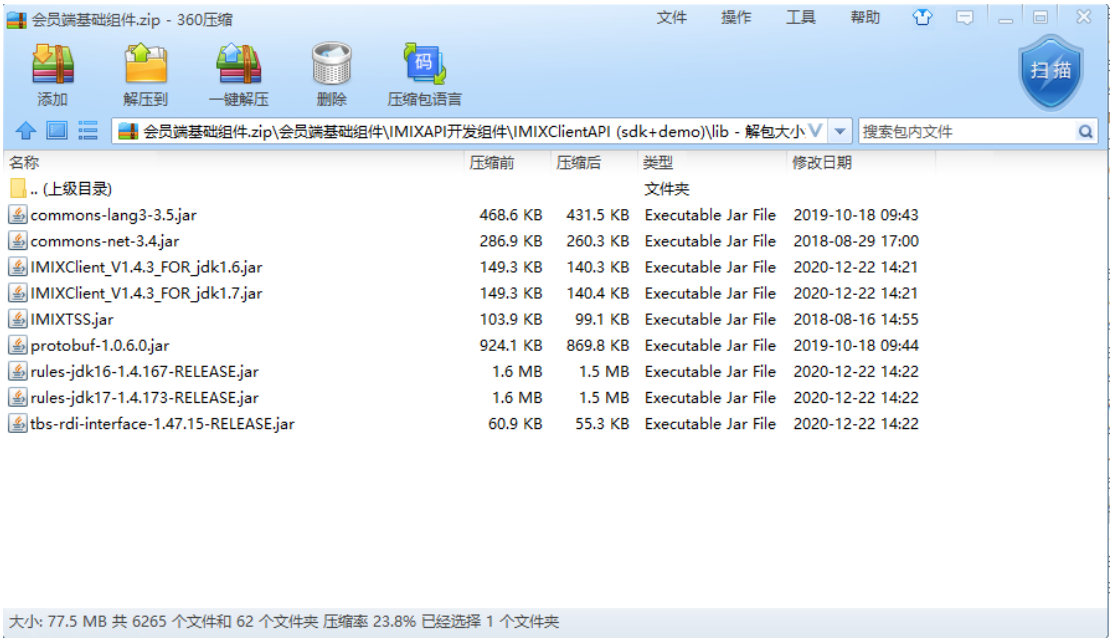
- **1、替换协议包：**协议包中有依赖的 jar 包，需要引入到自己的进程中才能正常使用，包目录如下图，会员端基础组件\1.149.9-2.14.0 兼容包\IMIXProtocol1and2.jar



  - IMIX 协议包，须放在接口进程的 lib 目录下

- **2、替换开发包：**
  - 开发包目录下的所有包都要放到接口进程的 lib 目录。

开发包目录为：会员端基础组件\IMIXAPI 开发组件\IMIXClientAPI (sdk+demo)\lib。具体截图见下图：



- **3、 删除旧版本开发包**
  - 替换后请将原 jar 包删除。

注意事项；” rules-jdk16-**-RELEASE.jar”,” rules-jdk17-**-RELEASE.jar” (**表示具体版本号)和” ImixClientAPI_**_FOR_jdk1.6.jar”,” ImixClientAPI_**_FOR_jdk1.7.jar”(**表示具体版本号)存在两个 jdk 版本 jar 包,请根据自己进程实际选择对应版本。

2.5 IMIX 基础组件环境

2.5.1 软件环境

OS	Software	Notes
Any that support JVM	JDK1.5.0 32 位	用户可以根据安装的 JDK 版本来选择使用 IMIXClient 以及 IMIXAgent 相对应的版本

2.5.2 JVM 参数

推荐以下 JVM 参数配置：

Paramaters	Notes
-Xms256m	set initial Java heap size
-Xmx512m	set maximum Java heap size

2.5.3 硬件环境

OS	Hard Disk	Memory	Notes
Any that support JVM	>= 40G	>= 1G	

2.6 开发 IMIX Client 应用程序

2.6.1 初始化

2.6.2 登录

2.6.3 交换消息

构造并发送消息.（实际消息以开发指引为准）

从回调中接收消息：

```
public class DemoListener extends MessageCracker implements Listener {
    public void fromApp(Message message, IMIXSession imixSession){
        System.out.print("fromApp:");
        System.out.println("imixSession:" + imixSession);
        System.out.println("message:" + message);
    }

public void onLogon(ImixSession imixSession) {
System.out.println("==============onLogon()================");
System.out.println("imixSession:" + imixSession);
System.out.println("====================================");
    }

public void onLogout(ImixSession imixSession) {
System.out.println("==============onLogout()===============");
System.out.println("imixSession:" + imixSession);
System.out.println("====================================");
    }

public void toApp(Message message, ImixSession imixSession) {
System.out.println("==============toApp()=================");
System.out.println("imixSession:" + imixSession);
```

```
System.out.println("==================================");
    }

public void fromAdmin(Message message, ImixSession imixSession) {
System.out.println("==============fromAdmin()==============");
System.out.println("imixSession:" + imixSession);
System.out.println("==================================");
    }

public void onError(ImixSession imixSession, int type) {
System.out.println("==============onError()================");
if (Listener.CONNECTION_FAILURE == type){
System.out.println("connection failure");
System.out.println("==================================");
        }
    }

public    void    onMessage(QueryReject    queryReject)    throws    FieldNotFound,
UnsupportedMessageType, IncorrectTagValue
    {
        //process the message here
    }
}
```

在这个例子里面可以在相应的 *onMessage* 方法中完成对 *QueryReject* 消息的处理。
其他的相关回调：

```
    public void onLogon(IMIXSession imixSession) {
        System.out.print("onLogon:");
        System.out.println("imixSession:" + imixSession);
    }
```

当某个 session logon 成功后被调用

```
    public void onLogout(IMIXSession imixSession) {
        System.out.print("onLogout:");
        System.out.println("imixSession:" + imixSession);
    }
```

当某个 session logout 后被调用

```
    public void toApp(Message message, IMIXSession imixSession) {
        System.out.print("toApp:");
        System.out.println("imixSession:" + imixSession);
        System.out.println("message:" + message);
    }
}
```

当有应用类（application）消息被发送时调用

2.6.4 登出

2.7 使用协议库解析 IMIX 消息

2.7.1 消息格式简介

IMIX 消息分为消息头（Header）、消息体（Body）、消息尾（Trailer）三部分，消息头（Header）主要承载消息传输协议本身所需要的一些信息；消息体（Body）主要承载消息内容本身；消息尾（Trailer）主要负责检验等工作。

为保证消息被正确传输，域“8、9、10、34、35、49、52、56”在“IMIX1.0”协议中为必需域，在消息头（Header）、消息尾（Trailer）中使用。

8	BeginString
9	BodyLength
35	MsgType
49	SenderCompID
56	TargetCompID
34	MsgSeqNum
52	SendingTime
10	CheckSum

域“43、50、57、89、90、91、93、97、115、116、122、128、129、142、143、144、145、212、213、347、369、627、628、629、630、1128、1129”在 IMIX 协议中作为可选域，在消息头（Header）、消息尾（Trailer）中使用，来保证消息被正确、安全地在多方之间传输。

消息头（Header）中的 MsgType 域（域 35）被用来设定消息类型；其中， "0"、"1"、"2"、"3"、"4"、"5"、"A" 六种消息类型被用来作为 Admin 消息，以保证消息通讯。其他消息类型均为 App 消息。

2.7.2 IMIX 消息的构造

API提供下列方法来构造消息：

方法一：	`Message message = new Message();`
	`NewOrderSingle newOrderSingle1 = newNewOrderSingle();`

	`NewOrderSingle newOrderSingle2 =`
	`newNewOrderSingle(newClOrdID("123"),newSide(Side.BUY),`
	`newTransactTime(),newOrdType('2'));`

方法二：	Message message = new Message(String string);
方法三：	Message message = Message.createMessage(String beginString, String msgType);

API提供下列方法来得到消息头（Header）和消息尾（Trailer）：

消息头：	message.getHeader();
消息尾：	message.getTrailer();

API提供下列 2 类方式来构造域：

方法一：	newStringField(Field_Tag, String_Value);
	newCharField (Field_Tag, Char_Value);
	newBooleanField (Field_Tag, Boolean_Value);
	newIntField(Field_Tag, Int_Value);
	new DoubleField(Field_Tag, Double_Value);

方法二：	newStringField(Field_Tag);
	newCharField (Field_Tag);
	newBooleanField (Field_Tag);
	newIntField(Field_Tag);
	new DoubleField(Field_Tag);

	newClOrdID("123");
	newPrice();

提供下列方法来把一个具体的域添加到一条消息中去：

方法一：	Message.setField(new XXField(Field_Tag, XX_Value));
方法二：	message.setField(newText("Cancel My Order!"));
	newOrderSingle.set(newPrice(1));
方法三：	String string = "Field_Tag=value\001" + "Field_Tag=value\001";

设置group的方式与设置消息体（Body）的方式相似，并提供下列方法添加Group到一条消息中去：

方法一：	Group group = createGroup(String beginString, String msgType, int correspondingFieldID);

方法二：	Group group = new Group（int field, int delim）;
	message.addGroup(group);

	MarketDataSnapshotFullRefresh.NoMDEntries group =
	new MarketDataSnapshotFullRefresh.NoMDEntries();
	group.set(newMDEntryType('0'));
	group.set(newMDEntryPx(12.32));
	group.set(newMDEntrySize(100));
	group.set(newOrderID("ORDERID"));
	message.addGroup(group);

对于接收到的消息，可以提供下列两种方式分别从消息头（Header）、消息体（Body）、消息尾（Trailer）中取出具体的域和值(包括group)：一是根据消息中应包含的具体的域取值；二是用迭代的方法取值。

	public void fromApp(iMixMessage message, Operator operator)
	{
	messageCrack(message);
	}

方法一：	onMessage(NewOrderSingle order, Operator operator)
	{
	Symbol symbol =newSymbol();
	Side side =newSide();
	OrdType ordType =newOrdType();
	OrderQty orderQty =newOrderQty();
	Price price =newPrice();
	ClOrdID clOrdID =newClOrdID();
	
	order.get(ordType);
	order.get(symbol);
	order.get(side);
	order.get(orderQty);
	order.getField(price);
	order.get(clOrdID);
	}

	MarketDataSnapshotFullRefresh.NoMDEntries group =
	new MarketDataSnapshotFullRefresh.NoMDEntries();
	
	NoMDEntries noMDEntries = new NoMDEntries();
	intGroupNum = message.get(noMDEntries).getValue();
	
	MDEntryType MDEntryType = new MDEntryType();
	MDEntryPx MDEntryPx = new MDEntryPx();

	MDEntrySize MDEntrySize = new MDEntrySize();
	OrderID orderID = new OrderID();
	
	for(int i = 1; i <= GroupNum; i++){
	message.getGroup(i, group);
	group.get(MDEntryType).getValue();
	group.get(MDEntryPx).getValue();
	group.get(MDEntrySize).getValue();
	group.get(orderID).getValue();
	}

方法二：	Iterator fieldIterator = fieldMap.iterator();
	while(fieldIterator.hasNext()) {
	Field field = (Field) fieldIterator.next();
	if(!isGroupCountField(dd, field)) {
	String value = fieldMap.getString(field.getTag());
	if(dd.hasFieldValue(field.getTag())) {
	value = dd.getValueName(field.getTag(),
	fieldMap.getString(field.getTag())) +" ("+ value +")";
	}
	System.*out*.println(prefix +
	dd.getFieldName(field.getTag()) +": "+ value);
	}
	}
	
	Iterator groupsKeys = fieldMap.groupKeyIterator();
	while (groupsKeys.hasNext()) {
	intgroupCountTag=((Integer)groupsKeys.next()).intValue();
	System.*out*.println(prefix + dd.getFieldName(groupCountTag)
	+": count = "+ fieldMap.getInt(groupCountTag));
	Group g =new Group(groupCountTag, 0);
	inti = 1;
	while(fieldMap.hasGroup(i, groupCountTag)) {
	if(i > 1) {
	System.out.println(prefix +" ----");
	}
	fieldMap.getGroup(i, g);
	printFieldMap(prefix +" ",dd, msgType, g);
	i++;
	}
	}

2.7.3 示例一

Initiator

```
//construt an order -- method 1:
  Message msg1 = new Message();
    Header header1 = msg1.getHeader();
    header1.setField(new MsgType("D"));
    //Message msg1 = Message.createMessage("IMIX.1.0", "D");
    msg1.setField(new ClOrdID("321"));
    msg1.setField(new Symbol("LNUX"));
    msg1.setField(new Side(Side.BUY));
    msg1.setField(new TransactTime());
    msg1.setField(new OrdType('2'));
    msg1.setField(new OrderQty(1));
    msg1.setField(new HandlInst('1'));
    msg1.setField(new Price(1));
msg1.setField(new StringField(58, "Cancel My Order!"));
    msg1.getHeader().setString(DeliverToCompID.FIELD, "CFETS");
```

Acceptor

```
public void onMessage(NewOrderSingle order)
throws FieldNotFound, UnsupportedMessageType, IncorrectTagValue {
        Symbol symbol = new Symbol();
        Side side = new Side();
        OrdType ordType = new OrdType();
        OrderQty orderQty = new OrderQty();
        Price price = new Price();
        ClOrdID clOrdID = new ClOrdID();

order.get(ordType);
order.get(symbol);
order.get(side);
order.get(orderQty);
order.get(price);
order.get(clOrdID);

        OrderID orderID = new OrderID("OID1");
        ExecID execID = new ExecID("EID1");

        ExecutionReport executionReport = new ExecutionReport(orderID,
execID, new ExecType(ExecType.FILL),
new OrdStatus(OrdStatus.FILLED), side,
new LeavesQty(0), new CumQty(orderQty.getValue()),
new AvgPx(price.getValue()));
```

```
executionReport.set(clOrdID);
executionReport.set(symbol);
executionReport.set(orderQty);
executionReport.set(new LastQty(orderQty.getValue()));
executionReport.set(new LastPx(price.getValue()));
executionReport.getHeader().setString(DeliverToCompID.FIELD,
order.getHeader().getString(OnBehalfOfCompID.FIELD));
executionReport.getHeader().setString(DeliverToSubID.FIELD,
order.getHeader().getString(OnBehalfOfSubID.FIELD));

        Operator deliverToOperator;

        String deliverToCompID =
order.getHeader().getString(DeliverToCompID.FIELD);
if(deliverToCompID.equals("CFETS")){
deliverToOperator = Operator.lookupOperator
                    (order.getHeader().getString(OnBehalfOfSubID.FIELD)+
order.getHeader().getString(OnBehalfOfCompID.FIELD));;
        }
else{
deliverToOperator = Operator.lookupOperator
                    (order.getHeader().getString(DeliverToSubID.FIELD)+
deliverToCompID);
        }

if (deliverToOperator != null && deliverToOperator.isLogon()) {
deliverToOperator.send(executionReport);
        } else {
throw new imix.RuntimeError("the DeliverToSubID operator is
not logon");
        }
}
```

Display

```
<20070907-09:33:57,                    iMIX.1.0:CFETS->H1,                    incoming>
(8=iMIX.1.0_9=164_35=D_34=3_49=H1_52=20070907-09:33:57.441_56=CFETS_115=MB1_116
=Sub001_128=CFETS_11=321_21=1_38=1_40=2_44=1_54=1_55=LNUX_58=Cancel          My
Order!_60=20070907-09:33:57.096_10=067_)

<20070907-09:33:57,                    iMIX.1.0:CFETS->H1,                    outgoing>
(8=iMIX.1.0_9=146_35=8_34=3_49=CFETS_52=20070907-09:33:57.566_56=H1_128=MB1_129
=Sub001_6=1_11=321_14=1_17=EID1_31=1_32=1_37=OID1_38=1_39=2_54=1_55=LNUX_150=2_
151=0_10=231_)
```

2.7.3 示例二

Initiator

```
//construt a message with component and group -- method 1
    MarketDataSnapshotFullRefresh msg2 =
        new MarketDataSnapshotFullRefresh();

    Instrument component = new Instrument(new Symbol("[N/A]"));
msg2.set(component);
    MarketDataSnapshotFullRefresh.NoMDEntries group =
        new MarketDataSnapshotFullRefresh.NoMDEntries();

    group.set(new MDEntryType('0'));
    group.set(new MDEntryPx(12.32));
    group.set(new MDEntrySize(100));
    group.set(new OrderID("ORDERID"));
    msg2.addGroup(group);

    group.set(new MDEntryType('1'));
    group.set(new MDEntryPx(12.33));
    group.set(new MDEntrySize(200));
    group.set(new OrderID("ORDERID"));
    msg2.addGroup(group);

    msg2.getHeader().setString(DeliverToCompID.FIELD, "CFETS");
```

Acceptor

```
public void onMessage(MarketDataSnapshotFullRefresh
message)throws FieldNotFound,
UnsupportedMessageType, IncorrectTagValue {

NoMDEntries noMDEntries = new NoMDEntries();
        intGroupNum = message.get(noMDEntries).getValue();

Symbol value = new Symbol();
        Instrument component = new Instrument();
        String g = message.get(component).get(value).getValue();

        System.out.println("Symbol: " + g);
        System.out.println("noMDEntries: " + GroupNum);

        MarketDataSnapshotFullRefresh.NoMDEntries group =
```

```
            new MarketDataSnapshotFullRefresh.NoMDEntries();
        MDEntryType MDEntryType = new MDEntryType();
        MDEntryPx MDEntryPx = new MDEntryPx();
        MDEntrySize MDEntrySize = new MDEntrySize();
        OrderID orderID = new OrderID();

        char b;
        double c;
        double d;
        String e;

for(int i = 1; i <= GroupNum; i++){
        message.getGroup(i, group);
          b = group.get(MDEntryType).getValue();
          c = group.get(MDEntryPx).getValue();
          d = group.get(MDEntrySize).getValue();
          e = group.get(orderID).getValue();

        System.out.println("noMDEntries: " + GroupNum + "-" + i);
        System.out.println("MDEntryType: " + b);
        System.out.println("MDEntryPx: " + c);
        System.out.println("MDEntrySize: " + d);
        System.out.println("orderID: " + e);
}
    }
```

Display

```
Symbol: [N/A]
noMDEntries: 2
noMDEntries: 2-1
MDEntryType: 0
MDEntryPx: 12.32
MDEntrySize: 100.0
orderID: ORDERID
noMDEntries: 2-2
MDEntryType: 1
MDEntryPx: 12.33
MDEntrySize: 200.0
orderID: ORDERID
```

2.7.5 示例三

Initiator

```
//construt a message from string -- method 2:
```

```
DataDictionary dd = new DataDictionary("iMIX10.xml");
        Message msg3 =
new Message("8=iMIX.1.0\0019=247\00135=s\00134=5\001"
              + "49=MB1\00150=Sub001\52=20060319-09:08:20.881\001"
              + "56=H1-Sub001\001128=CFETS\001"
              + "22=8\00140=2\00144=9\00148=ABC\00155=ABC\001"
              + "60=20060319-09:08:19\001548=184214\001549=2\001"
              + "550=0\001552=2\00154=1\00111=001\001453=2\001"
              + "448=8\001447=D\001452=4\001448=AAA35777\001"
              + "447=D\001452=3\00138=9\00154=2\00111=002\001453=2\001"
              + "448=8\001447=D\001452=4\001448=aaa\001447=D\001"
              + "452=3\00138=9\00110=139\001", dd);
```

Acceptor

```
 //receive message2
public void onMessage(NewOrderCross message)
throws FieldNotFound, UnsupportedMessageType, IncorrectTagValue
   {
    try {
       DataDictionary dd = new DataDictionary("iMIX10.xml");
       String msgType = message.getHeader().getString(MsgType.FIELD);
    printFieldMap("", dd, msgType, message.getHeader());
    printFieldMap("", dd, msgType, message);
    printFieldMap("", dd, msgType, message.getTrailer());
    }
    catch (ConfigError e) {
e.printStackTrace();
    }
   }

private void printFieldMap(String prefix, DataDictionary dd, String
msgType, FieldMap fieldMap) throws FieldNotFound {

     Iterator fieldIterator = fieldMap.iterator();
while (fieldIterator.hasNext()) {
        Field field = (Field) fieldIterator.next();
if (!isGroupCountField(dd, field)) {
           String value = fieldMap.getString(field.getTag());
if (dd.hasFieldValue(field.getTag())) {
value = dd.getValueName(field.getTag(),
fieldMap.getString(field.getTag())) + " (" + value + ")";
           }
System.out.println(prefix + dd.getFieldName(field.getTag())
                              + ": " + value);
```

```
        }
    }

    Iterator groupsKeys = fieldMap.groupKeyIterator();
while (groupsKeys.hasNext()) {
int groupCountTag = ((Integer) groupsKeys.next()).intValue();
System.out.println(prefix + dd.getFieldName(groupCountTag) +
                ": count = " + fieldMap.getInt(groupCountTag));
          Group g = new Group(groupCountTag, 0);
          //Group g = Group.createGroup("iMIX.1.0","s",groupCountTag);
int i = 1;
while (fieldMap.hasGroup(i, groupCountTag)) {
if (i > 1) {
System.out.println(prefix + "  ----");
                }
fieldMap.getGroup(i, g);
printFieldMap(prefix + "  ", dd, msgType, g);
i++;
          }
       }
}

private boolean isGroupCountField(DataDictionary dd, Field field) {
return dd.getFieldTypeEnum(field.getTag()) == FieldType.NumInGroup;
    }
```

Display

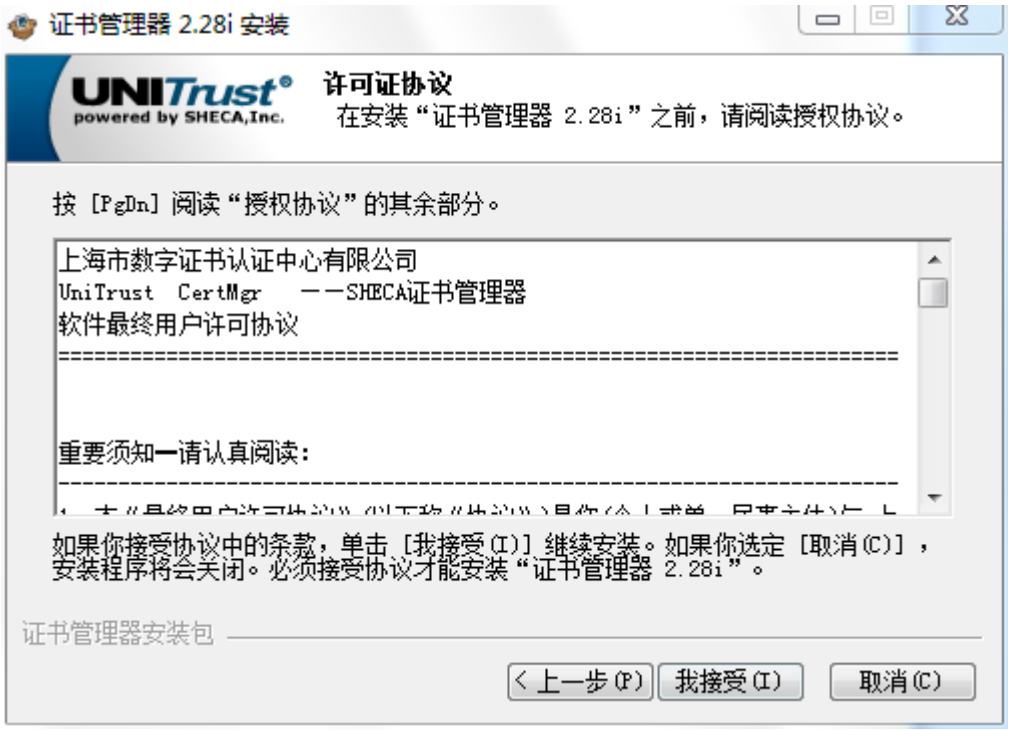
```
BeginString: iMIX.1.0
BodyLength: 285
MsgSeqNum: 5
MsgType: NewOrderCross (s)
SenderCompID: H1
SendingTime: 20070907-10:14:18.504
TargetCompID: CFETS
OnBehalfOfCompID: MB1
OnBehalfOfSubID: Sub001
DeliverToCompID: CFETS
SecurityIDSource: EXCHANGE_SYMBOL (8)
OrdType: LIMIT (2)
Price: 9
SecurityID: ABC
Symbol: ABC
TransactTime: 20060319-09:08:19
CrossID: 184214
```

```
CrossType: CROSS_TRADE_WHICH_IS_EXECUTED_PARTIALLY_AND_THE_REST_IS_CANCELLED (2)
CrossPrioritization: NONE (0)
NoSides: count = 2
  ClOrdID: 001
  OrderQty: 9
  Side: BUY (1)
  NoPartyIDs: count = 2
    PartyIDSource: PROPRIETARY_CUSTOM_CODE (D)
    PartyID: 8
    PartyRole: CLEARING_FIRM (4)
    ----
    PartyIDSource: PROPRIETARY_CUSTOM_CODE (D)
    PartyID: AAA35777
    PartyRole: CLIENT_ID (3)
  ----
  ClOrdID: 002
  OrderQty: 9
  Side: SELL (2)
  NoPartyIDs: count = 2
    PartyIDSource: PROPRIETARY_CUSTOM_CODE (D)
    PartyID: 8
    PartyRole: CLEARING_FIRM (4)
    ----
    PartyIDSource: PROPRIETARY_CUSTOM_CODE (D)
    PartyID: aaa
    PartyRole: CLIENT_ID (3)
CheckSum: 088
```

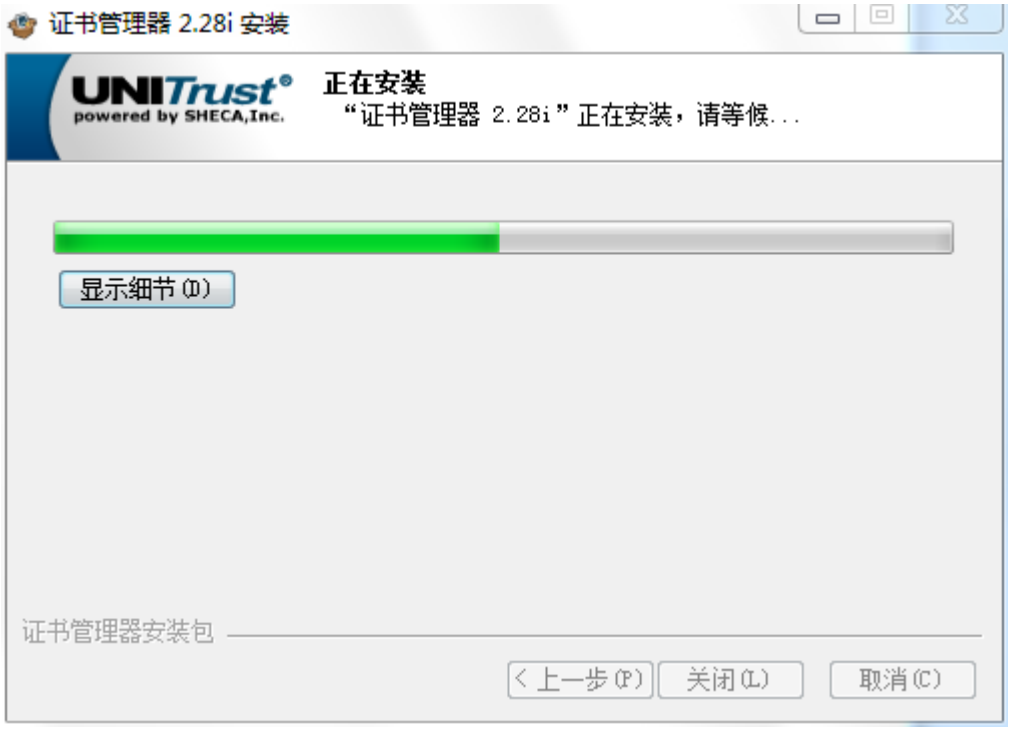
附录：

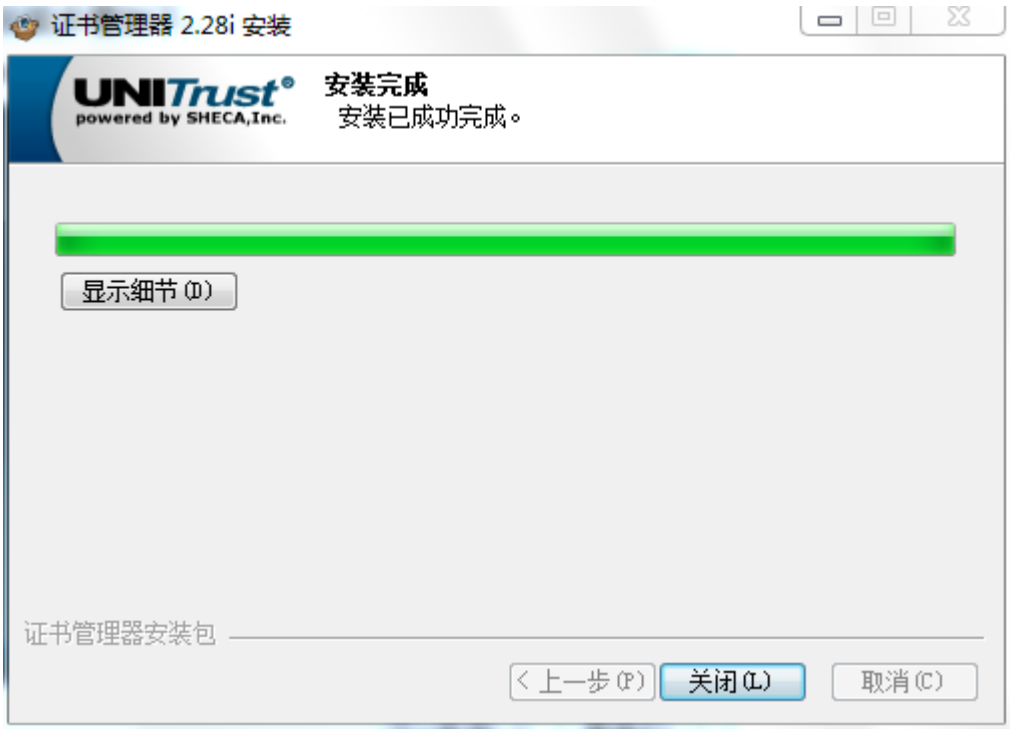
Windows 下的操作如下：

安装数字证书步骤如下：




证书管理器 2.28i 安装
UNITrust
powered by SHECA,Inc.
选择安装位置
选择“证书管理器 2.28i”的安装文件夹。
Setup 将安装 证书管理器 2.28i 在下列文件夹。要安装到不同文件夹，单击 [浏览(B)] 并选择其他的文件夹。 单击 [安装(I)] 开始安装进程。
目标文件夹
C:\Program Files (x86)\UniTrust
浏览(B)...
所需空间: 4.8MB
可用空间: 396.9GB
证书管理器安装包
< 上一步(P)
安装(I)
取消(C)

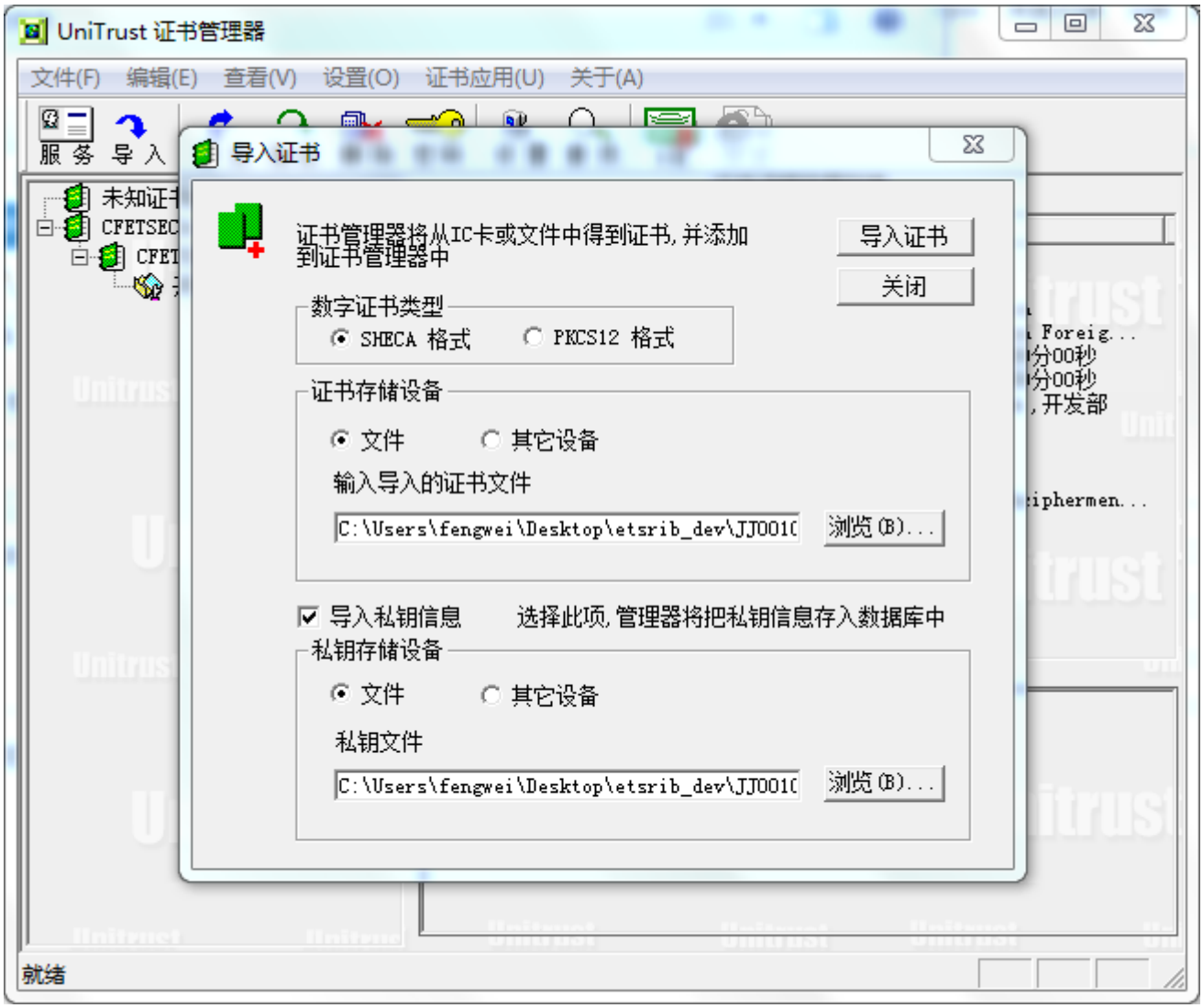

证书管理器 2.28i 安装
UNITrust
powered by SHECA,Inc.
正在安装
“证书管理器 2.28i”正在安装，请等候...
显示细节(D)
证书管理器安装包
< 上一步(P)
关闭(L)
取消(C)



安装完成后使用管理器转换 der 证书文件为 jks 文件

1 打开工具后，选择导入

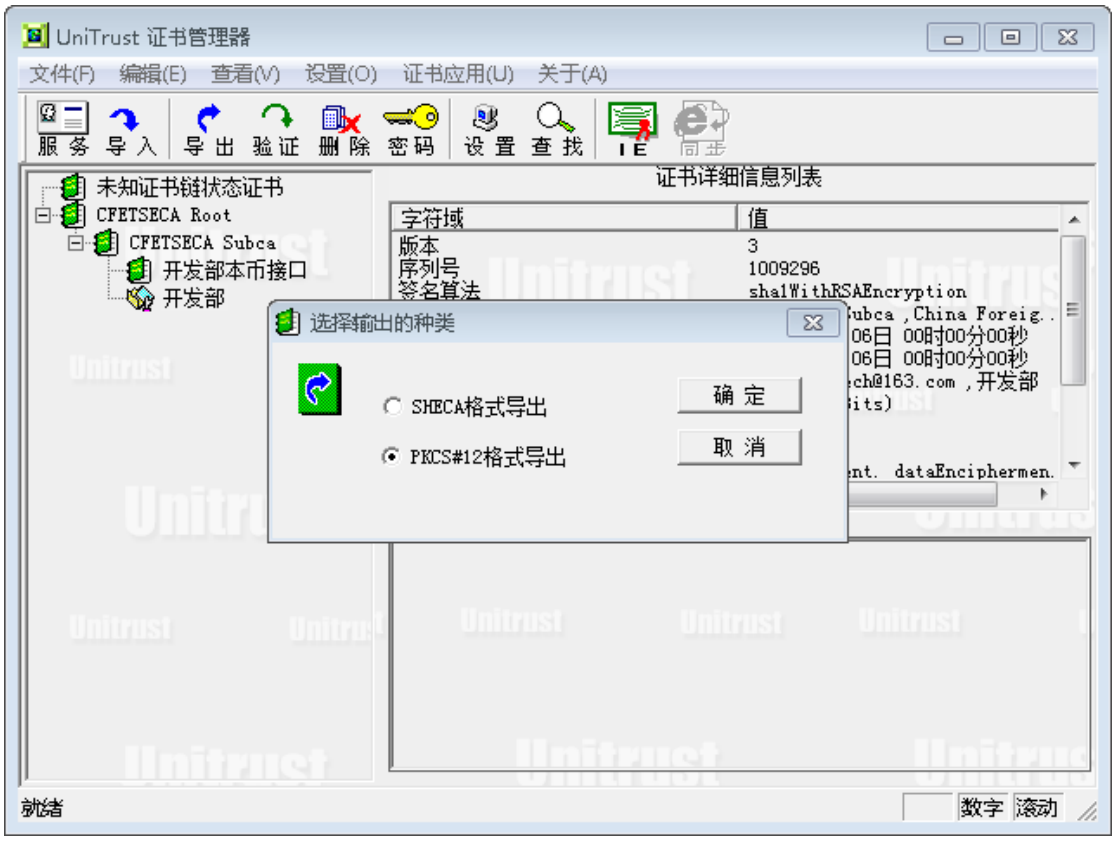
2 浏览，选择文件



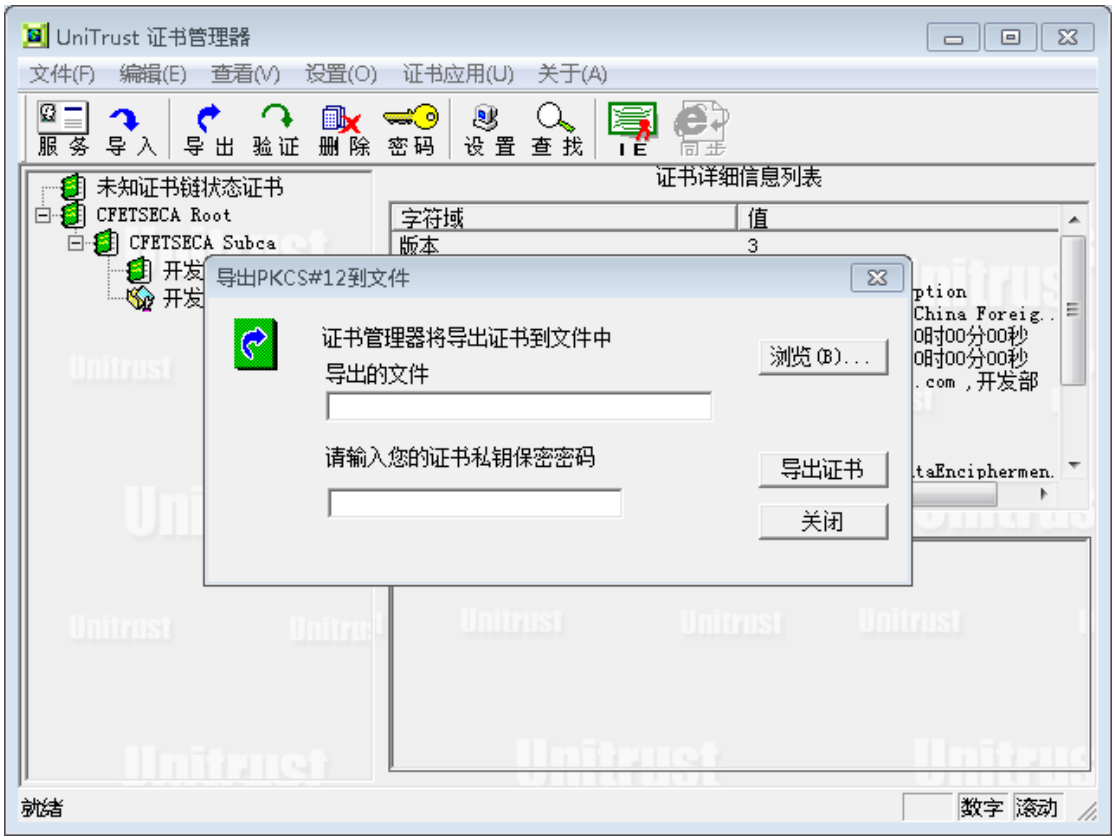
3 选择导入证书

4 输入密码后确定导入。

5 导入后选择导出，选择 PKCS#12 格式导出，然后点击确定



6.先选择浏览确定导出 P12 文件名和地址，然后输入密码，点击导出证书



7 到 jdk 安装目录下找到 bin 文件夹，bin 下有 keytool.exe,执行命令：

```
keytool -importkeystore -srckeystore UserCert.p12 -srcstoretype PKCS12
-deststoretype JKS -destkeystore UserCert.jks
```

（黄色为 p12 和要导出的 jks 文件的指定文件名和路径）。
回车后，根据提示操作。