jingtum-lib-csharp v0.8.0

设计说明书

# 概述

jingtum-lib是用于和jingtum区块链进行交互的基础库。已经有nodejs版的实现，现在需要支持.NET平台。功能参照nodejs v2.0.1，实现完全一样的功能。

说简单点就是：

1. 翻译nodejs各文件
2. 找到.net对应的nodejs中使用到的各模块

# 前期准备

## 阶段一：熟悉功能

通过阅读nodejs的接口说明书、API接口介绍、和源码docs.md，熟悉lib提供的各功能。

包括：

1. Wallet的生成
2. 服务器和账号相关信息的获取
3. 事务的提交（服务端签名和本地签名）
4. 账号和货币对相关事件的订阅

## 阶段二：通读nodejs代码

读取nodejs的代码，熟悉其基本设计结构以及各功能的实现方式。底层通过WS协议和jingtum系统交互，发送和接收的数据都是json格式。请求分为两种类型，Request和Transaction。Request用于请求信息，对所有用户都是公开的。Transaction需要设置secret才能提交，为了保护私钥安全，具有本地签名功能。

## 阶段三：熟悉概念

在阅读了文档和源码后，对一些概念还是不清楚。而理清概念对实现功能是必须的。

材料包括：

1. https://github.com/swtcpro/jingtum-lib-nodejs/blob/master/jingtum-lib/docs.md
2. jingtum-lib -v2.0.1.pdf
3. 井通区块链API接口介绍01-账号类+支付类+账本类.pdf
4. 井通区块链API接口介绍02-挂单类+**交易记录**+**交易效果**.pdf
5. <http://developer.jingtum.com/> （关系类交易）

### 货币

分为两种，基础货币和通证。基础货币就是SWT，用字符串表示数量，在和jingtum交互的时候需要乘以1000000并取整。通证由currency和issuer定义。Currency可以是3到6位的字符串，或者34位长的jingtum地址，issuer是发行方的jingtum地址。

### 本地签名

为了保护私钥安全，需要将信息在本地进行签名，然后将签名后的信息传到jingtum系统。签名的过程是：

1. 整理数据，将基础货币除以1000000，设置公钥。
2. 将数据以key/value对的形式序列化成二进制。
3. 将序列化结果取hash512。
4. 对hash进行ECDSA签名，并将签名结果放到数据中。
5. 对更新后的数据再进行序列化。
6. 将序列化的结果发送到jingtum系统。

### 事务的效果

每个事务都会有对应的影响效果。事务的影响信息不会在事务的基本信息中体现，而是在影响节点中提现出来。

比如转账就有如下效果：

1. 转账方资金减少
2. 接收方资金增加
3. jingtum方收取手续费

挂单成交有如下效果：

1. 创建了一个挂单节点
2. 挂单方资金变化
3. 被动成交方资金变化
4. 被动成交方挂点节点变化
5. jingtum方收取手续费

# 技术验证

有nodejs作为参照，在.net中实现应该不是什么难题。但是nodejs提供了丰富的第三方库，需要验证在.net中是否都可以实现。

## WebSocket

研究了WebSocket4Net和WebSocket-Sharp，都没有问题。

## Json序列化和反序列化

采用广泛使用的Newtonsoft.Json。

## 公钥私钥的生成和验证，ECDSA签名

使用轻量级的BouncyCastle。实现借鉴了<https://github.com/jingtum/jingtum-dotnet-sdk>。

# 设计

## 主要类结构

和Nodejs一样的设计。

* Remote类：提供公共API。
* Server类：使用Websocket和系统交互。
* Request类：收集参数，发送查询请求。
* Transaction类：收集参数，发送事务请求。
* Account类：监听Transactions事件并分发和账号相关的事务。
* OrderBook类：监听Transactions事件并分发和货币对相关的事务。

## 数据处理

由于发送和接收数据是多线程处理的，所有返回结果通过回调函数获取。获取的数据包括：

1. 系统返回的原始json数据。
2. 错误信息。包括本地的参数检查错误，和系统返回的错误。
3. 对原始数据的包装对象。为了让用户方便的获取返回数据及部分信息。将json数据解析成包装类。

## 异步调用

为了方便多线程同步操作，Request和Transaction的Submit方法应该提供异步调用。

## .NET Standard支持

在下一版本中应该支持夸平台的.NET Standard。现在的版本是2.0，应该尽可能的支持较早的版本。