jingtum-lib-oc使用说明

V0.1.0

**版本历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **简介** | **作者** | **日期** |
| 0.1.0 | 出版主要接口说明 | Jerry | 2018/8/21 |
|  |  |  |  |

本文档介绍jingtum-lib在iOS平台上的如何使用，另有接口的详细文档。

# 安装

通过CocoaPods安装。

pod "jingtum-lib-objectc"

# 项目文件结构

jingtum-lib库基于websocket协议跟底层交互，其中websocket封装到Server类中，Server类是一个内部类；Server类封装在Remote类中，Remote类提供对外接口并可创建两类对象：Get方式请求的Request对象和Post方式请求的Transaction对象，这两类对象都通过Submit()方法提交数据到底层。文件结构图如下：

Remote类

Request对象（Get请求）

WebSocket

通讯协议

Server

内部类

Transaction对象（Post请求）

# 创建钱包

首先通过CocoaPods添加jingtum-lib引用到项目中，代码中添加

#import <Wallet.h>

方法一：

Wallet \*wallet1 = [Wallet generate];

    NSLog(@"the wallet is %@", wallet1);

方法二：

Wallet \*wallet2 = [Wallet fromSecret:@"ss4EUqv9CqWtitJpwvGdDapE5GL9k"];

    NSLog(@"the wallet2 is %@", wallet2);

参数：

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| secret | string | 井通钱包私钥 |

# Remote类

Remote是跟井通底层交互最主要的类，它可以组装交易发送到底层、订阅事件及从底层拉取数据。其主要方法如下。

## 创建Remote类

通过构造函数传入井通底层服务地址并制定是否本地签名来创建Remote类。

Remote \*remote = [Remote instance];

    [remote connectWithURLString:@"ws://123.57.219.57:5020" local\_sign:true];

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| url | string | 井通底层服务地址 |
| localSign | bool | 交易是否以本地签名的方式发送给底层 |

## 创建连接

每个Remote对象都应该首先通过Connect方法手动连接底层，然后才可以请求底层数据。请求结果在回调函数callback中。

代码示例：

Remote \*remote = [Remote instance];

    [remote connectWithURLString:@"ws://123.57.219.57:5020" local\_sign:true];

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(SRWebSocketDidOpen) name:kWebSocketDidOpen object:nil];

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(SRWebSocketDidReceiveMsg:) name:kWebSocketdidReceiveMessage object:nil];

## 关闭连接

每个Remote对象可以手动通过Disconnect方法关闭连接。

代码示例：

[remote disconnect];

## 请求底层服务器信息

通过requestServerInfo方法获得井通底层的服务器信息，包含服务程序版本号Version、该服务器缓存的账本区间Ledgers、节点公钥Node、服务器当前状态State。其中服务器当前状态包含可提供服务状态full和验证节点状态proposing。

代码示例：

[remote requestServerInfo];

## 获取最新账本信息

通过requestLedgerClosed方法获得最新账本信息，包括区块高度(LedgerIndex)与区块hash(LedgerHash)。

代码示例：

[remote requestLedgerClosed];

## 获取某一账本具体信息

通过requestLedger方法获得某一账本的具体信息。

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| LedgerIndex | uint | 井通区块高度 |
| LedgerHash | string | 井通区块hash(与上面ledger\_index二选其一) |
| Transactions | bool | 是否返回账本上的交易记录hash，默认false |

代码示例：

NSMutableDictionary \*options = [[NSMutableDictionary alloc] init];

NSNumber \*ledger\_index = [NSNumber numberWithInt:483718];

NSNumber \*transactions = [NSNumber numberWithBool:YES];

[options setObject:ledger\_index forKey:@"ledger\_index"];

[options setObject:transactions forKey:@"transactions"];

[remote requestLedger:options];

## 查询某一交易具体信息

通过requestTx方法获得某一交易的具体信息。

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Hash | string | 交易hash |

代码示例：

[options setObject:@"A4C52EF5A3075BF6169BA0AC716BF26A989B97BEDE53DC3BA1C252CF1338E0C7" forKey:@"hash"];

[remote requestTx:options];

## 请求账号信息

通过requestAccountInfo方法获得某一账号的交易信息。

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Account | string | 井通钱包地址 |

代码示例：

[options setObject:@"jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" forKey:@"account"];

[remote requestAccountInfo:options];

## 获得账号可接收和发送的货币

通过equestAccountTums获得某一账号可发送和接收的货币种类。

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Account | string | 井通钱包地址 |

代码示例：

[options setObject:@"jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" forKey:@"account"];

[remote requestAccountTums:options];

## 获得账号关系

井通账户之间会建立各种不同的关系。这些关系由井通后台的关系（relations）机制来处理，

目前支持以下关系：信任(trust)、授权(authorize)、冻结(freeze)。

通过requestAccountRelations方法获得某一账号指定关系的信息。

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Account | string | 井通钱包地址 |
| Type | RelationType (enum) | 关系类型，支持Trust，Authorize，Freeze |

代码示例：

        [options setObject:@"jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" forKey:@"account"];

        [options setObject:@"trust" forKey:@"type"];

        [remote requestAccountRelations:options];

## 获得账号挂单

通过requestAccountOffers方法获得某一账号的挂单信息。

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Account | string | 井通钱包地址 |

代码示例：

        [options setObject:@"jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" forKey:@"account"];

        [remote requestAccountOffers:options];

## 获得账号交易列表

通过requestAccountTx方法获得某一账号的交易列表信息。

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Account | string | 井通钱包地址 |
| Limit | uint | 限定返回多少条记录，默认200 |

代码示例：

        [options setObject:@"jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" forKey:@"account"];

        [remote requestAccountTx:options];

## 获得市场挂单列表

通过requestOrderBook方法获得市场挂单列表信息。

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Gets | Amount | 挂单方获得的货币信息 |
| Pays | Amount | 挂单方支付的货币信息 |

代码示例：

        NSMutableDictionary \*gets = [[NSMutableDictionary alloc] init];

        NSMutableDictionary \*pays = [[NSMutableDictionary alloc] init];

        [gets setObject:@"SWT" forKey:@"currency"];

        [gets setObject:@"" forKey:@"issuer"];

        [pays setObject:@"CNY" forKey:@"currency"];

        [pays setObject:@"jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS" forKey:@"issuer"];

        [options setObject:gets forKey:@"gets"];

        [options setObject:pays forKey:@"pays"];

        NSNumber \*limit = [NSNumber numberWithInteger:2];

        [options setObject:limit forKey:@"limit"];

        [remote requestOrderBook:options];

## 支付

首先通过buildPaymentTx方法返回一个Transaction对象，然后通过SetSecret传入密钥，

AddMemo添加备注为可选项，最后通过Submit方法提交支付信息。

### 创建支付对象

Remote类的方法：

-(Transaction\*)buildPaymentTx:(NSDictionary\*)paramDic;

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Account | String | 发起账号 |
| To | String | 目标账号 |
| Amount | Amount | 支付金额 |
| Value | String | 支付数量 |
| Currency | String | 货币种类，三到六个字母或20字节的自定义货币 |
| Issuer | string | 货币发行方 |

### 传入密钥

[tx setSecret:@"sn37nYrQ6KPJvTFmaBYokS3FjXUWd"];

### 设置备注

[tx addMemo:@"给jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c支付0.5swt."];

### 提交支付

[tx submit];

示例代码：

[options setObject:@"jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" forKey:@"account"];

        [options setObject:@"jpKcDjvqT1BJZ6G674tvLhYdNPtwPDU6vD" forKey:@"to"];

    //    [options setObject:@"jpKcDjvqT1BJZ6G674tvLhYdNPtwPDU6vD" forKey:@"account"];

    //    [options setObject:@"jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" forKey:@"to"];

        NSMutableDictionary \*amount = [[NSMutableDictionary alloc] init];

        NSNumber \*value = [NSNumber numberWithFloat:2];

        [amount setObject:value forKey:@"value"];

        [amount setObject:@"SWT" forKey:@"currency"];

        [amount setObject:@" " forKey:@"issuer"];

        [options setObject:amount forKey:@"amount"];

        Transaction \*tx = [[Remote instance] buildPaymentTx:options];

        [tx setSecret:@"sn37nYrQ6KPJvTFmaBYokS3FjXUWd"];

    //    [tx setSecret:@"ssPstTqs7hTWXzDFj88Um9fZDeNUK"];

        [tx addMemo:@"给jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c支付0.5swt."];

        [tx submit];

## 设置关系

首先通过BuildRelationTx方法返回一个Transaction对象，然后通过SetSecret传入密钥，最后通过Submit方法提交支付信息。目前支持的关系类型：信任(trust)、授权(authorize)、冻结(freeze)。

### 创建关系对象

Remote类的方法：

-(Transaction\*)buildRelationTx:(NSDictionary\*)paramDic

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Account | String | 设置关系的源账号 |
| Target | String | 目标账号，授权和冻结才有 |
| Type | RelationType (enum) | 关系种类 |
| Limit | Amount | 关系金额 |
| Value | String | 支付数量 |
| Currency | String | 货币种类，三到六个字母或20字节的自定义货币 |
| Issuer | string | 货币发行方 |

### 传入密钥

[tx setSecret:@"sn37nYrQ6KPJvTFmaBYokS3FjXUWd"];

### 关系设置

[options setObject:@"jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" forKey:@"account"];

        [options setObject:@"jDUjqoDZLhzx4DCf6pvSivjkjgtRESY62c" forKey:@"target"];

        NSMutableDictionary \*limit = [[NSMutableDictionary alloc] init];

        [limit setObject:@"CCA" forKey:@"currency"];

        [limit setObject:@"0.112" forKey:@"value"];

        [limit setObject:@"js7M6x28mYDiZVJJtfJ84ydrv2PthY9W9u" forKey:@"issuer"];

        [options setObject:limit forKey:@"limit"];

        [options setObject:@"authorize" forKey:@"type"];

        Transaction \*tx = [[Remote instance] buildRelationTx:options];

        [tx setSecret:@"sn37nYrQ6KPJvTFmaBYokS3FjXUWd"];

        [tx submit];

## 设置账号属性

首先通过BuildAccountSetTx方法返回一个Transaction对象，然后通过SetSecret传入密钥，

最后通过Submit方法设置账号属性。目前支持的三类：property、delegate、signer。property

用于设置账号一般属性；delegate用于某账号设置委托账户；signer用于设置签名。

### 创建属性对象

Remote类的方法：

-(Transaction\*)buildAccountSetTx:(NSDictionary \*)paramDic

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Account | String | 设置属性的源账号 |
| Type | AccountSetType (enum) | 属性种类 |
| SetFlag | SetClearFlag (enum) | 属性编号 |

### 传入密钥

[tx setSecret:@"sn37nYrQ6KPJvTFmaBYokS3FjXUWd"];

### 属性设置

[options setObject:@"jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" forKey:@"account"];

        [options setObject:@"property" forKey:@"type"];

        Transaction \*tx = [[Remote instance] buildAccountSetTx:options];

        if (tx != nil) {

            // signerSet 会返回 nil

            [tx setSecret:@"sn37nYrQ6KPJvTFmaBYokS3FjXUWd"];

            [tx submit];

        }

## 挂单

首先通过BuildOfferCreateTx方法返回一个Transaction对象，然后通过SetSecret传入密

钥，最后通过Submit方法提交挂单。

### 创建挂单对象

Remote类的方法：

-(Transaction\*)buildOfferCreateTx:(NSDictionary \*)paramDic

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Account | String | 挂单方账号 |
| Type | OfferType (enum) | 挂单类型，可选两个值：Buy、Sell |
| TakerGets | Amount | 对方得到的，即挂单方支付的 |
| Value | String | 数量 |
| Currency | String | 货币种类 |
| Issuer | String | 货币发行方 |
| TakerPays | Amount | 对方支付的，即挂单方获得的 |
| Value | String | 数量 |
| Currency | String | 货币种类 |
| Issuer | String | 货币发行方 |

### 传入密钥

 [tx setSecret:@"sn37nYrQ6KPJvTFmaBYokS3FjXUWd"];

### 提交挂单

[options setObject:@"Sell" forKey:@"type"];

        [options setObject:@"jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" forKey:@"account"];

        NSMutableDictionary \*taker\_gets = [[NSMutableDictionary alloc] init];

        [taker\_gets setObject:@"CNY" forKey:@"currency"];

        [taker\_gets setObject:@"0.01" forKey:@"value"];

        [taker\_gets setObject:@"jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS" forKey:@"issuer"];

        NSMutableDictionary \*taker\_pays = [[NSMutableDictionary alloc] init];

        [taker\_pays setObject:@"SWT" forKey:@"currency"];

        [taker\_pays setObject:@"1" forKey:@"value"];

        [taker\_pays setObject:@"" forKey:@"issuer"];

        [options setObject:taker\_gets forKey:@"taker\_gets"];

        [options setObject:taker\_pays forKey:@"taker\_pays"];

        Transaction \*tx = [[Remote instance] buildOfferCreateTx:options];

        [tx setSecret:@"sn37nYrQ6KPJvTFmaBYokS3FjXUWd"];

        [tx submit];

## 取消挂单

首先通过BuildOfferCancelTx方法返回一个Transaction对象，然后通过SetSecret传入密

钥，最后通过Submit方法取消挂单。

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| Account | String | 挂单方账号 |
| Sequence | unit | 取消的单子号 |

### 创建取消挂单对象

Remote类的方法：

1. -(Transaction\*)buildOfferCancelTx:(NSDictionary\*)paramDic

### 传入密钥

[tx setSecret:@"sn37nYrQ6KPJvTFmaBYokS3FjXUWd"];

### 取消挂单

NSNumber \*sequence = [NSNumber numberWithInt:1936];

        [options setObject:sequence forKey:@"sequence"];

        [options setObject:@"jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ" forKey:@"account"];

        Transaction \*tx = [[Remote instance] buildOfferCancelTx:options];

        [tx setSecret:@"sn37nYrQ6KPJvTFmaBYokS3FjXUWd"];

        [tx submit];

# 底层常见错误附录

|  |  |
| --- | --- |
| **错误名称** | **说明** |
| tecCLAIM | Fee claimed. Sequence used. No action. |
| tecDIR\_FULL | Can not add entry to full directory. |
| tecFAILED\_PROCESSING | Failed to correctly process transaction. |
| tecINSUF\_RESERVE\_LINE | Insufficient reserve to add trust line. |
| tecINSUF\_RESERVE\_OFFER | Insufficient reserve to create offer. |
| tecNO\_DST | Destination does not exist. Send SWT to create it. |
| tecNO\_DST\_INSUF\_SWT | Destination does not exist. Too little SWT sent to create it. |
| tecNO\_LINE\_INSUF\_RESERVE | No such line. Too little reserve to create it. |
| tecNO\_LINE\_REDUNDANT | Can't set non-existent line to default. |
| tecPATH\_DRY | Path could not send partial amount. |
| tecPATH\_PARTIAL | Path could not send full amount. |
| tecMASTER\_DISABLED | Master key is disabled. |
| tecNO\_REGULAR\_KEY | Regular key is not set. |
| tecUNFUNDED | One of \_ADD, \_OFFER, or \_SEND. Deprecated. |
| tecUNFUNDED\_ADD | "Insufficient SWT balance for WalletAdd. |
| tecUNFUNDED\_OFFER | Insufficient balance to fund created offer. |
| tecUNFUNDED\_PAYMENT | Insufficient SWT balance to send. |
| tecOWNERS | Non-zero owner count. |
| tecNO\_ISSUER | Issuer account does not exist. |
| tecNO\_AUTH | Not authorized to hold asset. |
| tecNO\_LINE | No such line. |
| tecINSUFF\_FEE | Insufficient balance to pay fee. |
| tecFROZEN | Asset is frozen. |
| tecNO\_TARGET | Target account does not exist. |
| tecNO\_PERMISSION | No permission to perform requested operation. |
| tecNO\_ENTRY | No matching entry found. |
| tecINSUFFICIENT\_RESERVE | Insufficient reserve to complete requested operation. |
| tecNEED\_MASTER\_KEY | The operation requires the use of the Master Key. |
| tecDST\_TAG\_NEEDED | A destination tag is required. |
| tecINTERNAL | An internal error has occurred during processing. |
| tefALREADY | The exact transaction was already in this ledger. |
| tefBAD\_ADD\_AUTH | Not authorized to add account. |
| tefBAD\_AUTH | Transaction's private key is not authorized. |
| tefBAD\_LEDGER | Ledger in unexpected state. |
| tefCREATED | Can't add an already created account. |
| tefEXCEPTION | Unexpected program state. |
| tefFAILURE | Failed to apply. |
| tefINTERNAL | Internal error. |
| tefMASTER\_DISABLED | Master key is disabled. |
| tefMAX\_LEDGER | Ledger sequence too high. |
| tefNO\_AUTH\_REQUIRED | Auth is not required. |
| tefPAST\_SEQ | This sequence number has already past. |
| tefWRONG\_PRIOR | This previous transaction does not match. |
| telLOCAL\_ERROR | Local failure. |
| telBAD\_DOMAIN | Domain too long. |
| telBAD\_PATH\_COUNT | Malformed: Too many paths. |
| telBAD\_private\_KEY | private key too long. |
| telFAILED\_PROCESSING | Failed to correctly process transaction. |
| telINSUF\_FEE\_P | Fee insufficient. |
| telNO\_DST\_PARTIAL | Partial payment to create account not allowed. |
| telBLKLIST | Tx disable for blacklist. |
| telINSUF\_FUND | Fund insufficient. |
| temMALFORMED | Malformed transaction. |
| temBAD\_AMOUNT | Can only send positive amounts. |
| temBAD\_AUTH\_MASTER | Auth for unclaimed account needs correct master key. |
| temBAD\_CURRENCY | Malformed: Bad currency. |
| temBAD\_EXPIRATION | Malformed: Bad expiration. |
| temBAD\_FEE | Invalid fee, negative or not SWT. |
| temBAD\_ISSUER | Malformed: Bad issuer. |
| temBAD\_LIMIT | Limits must be non-negative. |
| temBAD\_QUORUM | Quorums must be non-negative. |
| temBAD\_WEIGHT | Weights must be non-negative. |
| temBAD\_OFFER | Malformed: Bad offer. |
| temBAD\_PATH | Malformed: Bad path. |
| temBAD\_PATH\_LOOP | Malformed: Loop in path. |
| temBAD\_SEND\_SWT\_LIMIT | Malformed: Limit quality is not allowed for SWT to SWT. |
| temBAD\_SEND\_SWT\_MAX | Malformed: Send max is not allowed for SWT to SWT. |
| temBAD\_SEND\_SWT\_NO\_DIRECT | Malformed: No Skywell direct is not allowed for SWT to SWT. |
| temBAD\_SEND\_SWT\_PARTIAL | Malformed: Partial payment is not allowed for SWT to SWT. |
| temBAD\_SEND\_SWT\_PATHS | Malformed: Paths are not allowed for SWT to SWT. |
| temBAD\_SEQUENCE | Malformed: Sequence is not in the past. |
| temBAD\_SIGNATURE | Malformed: Bad signature. |
| temBAD\_SRC\_ACCOUNT | Malformed: Bad source account. |
| temBAD\_TRANSFER\_RATE | Malformed: Transfer rate must be >= 1.0. |
| temDST\_IS\_SRC | Destination may not be source. |
| temDST\_NEEDED | Destination not specified. |
| temINVALID | The transaction is ill-formed. |
| temINVALID\_FLAG | The transaction has an invalid flag. |
| temREDUNDANT | Sends same currency to self. |
| temREDUNDANTSIGN | Add self as additional sign. |
| temSKYWELL\_EMPTY | PathSet with no paths. |
| temUNCERTAIN | In process of determining result. Never returned. |
| temUNKNOWN | The transaction requires logic that is not implemented yet. |
| temDISABLED | The transaction requires logic that is currently disabled. |
| temMULTIINIT | Contract code has multi init function. |
| terRETRY | Retry transaction. |
| terFUNDS\_SPENT | Can't set password, password set funds already spent. |
| terINSUF\_FEE\_B | Account balance can't pay fee. |
| terLAST | Process last. |
| terNO\_SKYWELL | Path does not permit rippling. |
| terNO\_ACCOUNT | The source account does not exist. |
| terNO\_AUTH | Not authorized to hold IOUs. |
| terNO\_LINE | "No such line. |
| terPRE\_SEQ | Missing/inapplicable prior transaction. |
| terOWNERS | Non-zero owner count. |
| tesSUCCESS | The transaction was applied. Only final in a validated ledger. |