合约SDK接入文档

1.0 兼容性

支持 iOS10.0 以上的系统。其他系统需要进行测试。

1.1 软硬件支持

硬件: iPhone。

开发环境:Xcode10.13 及以上版本。

2.0 合约SDK功能介绍

2.1 市场实时数据接入

2.2 K线实时数据接入

2.3交易功能接口接入

2.4资金划转功能接入

2.5个人合约资产数据接入

2.6 websocket 市场、交易、资产实时接入

3.0 集成SDK

cocoapods集成

1. 将 SDK包中的SLContractSDK.framework SLContractSDK.bundle拖入到到工程（后续添加到pod中）
2. pod ‘AFNetworking’

pod ‘SocketRocket’

Pod ‘Reachaility’

手动集成

1. 将 SDK 包中的 SLContractSDK.framework SLContractSDK.bundle拖入到到工程。(注意两个文件)

拖入 AFNetworking SocketRocket Reachability库

3.1 导入头文件

#import <SLContractSDK/SLContractSDK.h>

3.2 初始化

3.2.1 合约信息初始化

在 didFinishLaunchingWithOptions中初始化合约配置信息

AppID:对应服务端给接入方分配的appid（务必要传（目前传入对应app的bundleName，不然sdk不会初始化）

launchOption：

base\_host: 接入方提供HTTPHost

base\_Header: 接入方提供

[[SLSDK sharedInstance] sl\_startWithAppID:@"12" launchOption:@{@"base\_host": @"https://api.abcapp.vip/v1",

                                                                   @"host\_Header":@"Abc"

                                                                  } callBack:^(id result, NSError \*error) {

                                                                       if (error == nil) {

                                                                           NSLog(@"load success");

                                                                       } else {

                                                                           NSLog(@"load error");

                                                                       }

                                                                   }];

3.2.2 账户信息初始化

参考Demo 中的 SLLoginController，需要手动初始化一个 BTAccount

BTAccount \*account = [[BTAccount alloc] init];

// 账户 id, 如果没有, 可以不传

account.Uid = @"xxx";

// 从 forchain 系统申请而来的 secret\_key

account.Token = @"xxx";

// 从forchain 系统申请而来 api\_key

account.access\_key = @"xxx";

// 过期时间

account.expiredTs = @"xxx";

// 把账户赋值给 sdk

[[SLPlatformSDK sharedInstance] sl\_startWithAccountInfo:account];

3.2.3 合约市场

SLSDK类中数据接口可在未登录状态下调用

/\*\*

 \*  @brief      合约市场数据

 \*  finished    返回结果

 \*/

+ (void)sl\_loadFutureMarketData:(SLRequestCallBack)finished;

/\*\*

 \*  @brief      获取合约最新成交记录

 \*  @param      contract\_id  合约id

 \*  callBack    返回最新成交数组

 \*/

+ (void)sl\_loadFutureLatestDealWithContractID:(int64\_t)contract\_id

                                 callbackData:(SLDataCallBack)callBack;

/\*\*

 \*  @brief      获取合约深度

 \*  @param      contract\_id     合约id

 \*  @param      price           最新成交价

 \*  @param      count           数量

 \*  success     深度模型

 \*/

+ (void)sl\_loadFutureDepthWithContractID:(int64\_t)contract\_id

                                   price:(NSString \*)price

                                   count:(NSInteger)count

                                 success:(void (^)(BTDepthModel \*))success

                                 failure:(void (^)(NSError \*))failure;

/\*\*

 \*  @brief      获取保险基金记录

 \*  @param      contract\_id     合约id

 \*/

+ (void)sl\_loadRiskReservesWithContractID:(int64\_t)contract\_id

                                  success:(void (^)(NSArray <BTIndexDetailModel \*>\*result))success

                                  failure:(void (^)(id ))failure;

/\*\*

 \*  @brief      获取资金费率

 \*  @param      contract\_id     合约id

 \*/

+ (void)sl\_loadFundingrateWithContractID:(int64\_t)contract\_id

                                 success:(void (^)(NSArray <BTIndexDetailModel \*>\*result))success

                                 failure:(void (^)(id ))failure;

/\*\*

 \*  @brief      更新币种对应USD或者BTC等币的汇率

 \*/

+ (void)sl\_updataCoinPrice;

/\*\*

 \*  @brief      请求合约K线数据

 \*  @param      contract\_id     合约id

 \*  @param      startTime       开始时间

 \*  @param      endTime         结束时间

 \*  @param      unit            单位

 \*  @param      resolution      时间单位（M:分钟，H:小时， D:天）

 \*  success     回调合约模型数组

 \*  error       返回错误信息

 \*  @dicuss

 \*/

+ (void)sl\_loadFutureKLineDataWithContractID:(int64\_t)contract\_id

                                   startTime:(NSNumber \*)startTime

                                     endTime:(NSNumber \*)endTime

                                        unit:(NSString \*)unit

                                  resolution:(NSString \*)resolution

                                     success:(void (^)(NSArray <BTItemModel \*>\*lineData))success

                                     failure:(void (^)(id error))failure;

3.2.4 合约交易

SLPlatformSDK类中数据接口需要在传入 activeAccount 之后调用

1> 初始化SLPlatformSDK

// 传入当前账户信息

- (void)sl\_startWithAccountInfo:(BTAccount \*)account;

2> 初始化之后需要更新手续费配置，不同用户收取的手续费可能存在差异

- (void)sl\_reloadUserFeeConfig;

3> 合约交易请求接口数据模型

BTContract 合约基本信息模型

BTContractFeeConfigModel 合约手续费模型

BTContractRiskLimitModel 合约风险限额模型

BTContractsModel 合约信息数据模型

BTContractsOpenModel 合约开仓数据模型

BTContractRecordModel 合约交易记录模型

BTContractOrderModel 合约订单模型

BTCashBooksModel 资金流水模型

BTPositionModel 合约仓位模型

4> 合约交易数据请求接口

BTContractTool 类中有所有的合约交易的请求接口

1、需要首先开通合约账户

// 创建合约账户

+ (void)createContractAccountWithContractID:(int64\_t)contract\_id

                                    success:(void (^)(id result))success

                                    failure:(void (^)(id ))failure;

1. 创建合约订单

BTContractOrderModel 合约订单模型中有直接创建合约订单的方法

// 创建开仓订单

+ (instancetype)newContractOpenOrderWithContractId:(int64\_t)contract\_id

                 category:(BTContractOrderCategory)category

                      way:(BTContractOrderWay)way

                 openType:(BTPositionOpenType)open\_type

                 leverage:(NSString \*)leverage

                    price:(NSString \*)price

                      vol:(NSString \*)vol;

// 创建平仓订单

+ (instancetype)newContractCloseOrderWithContractId:(int64\_t)contract\_id

                                      category:(BTContractOrderCategory)category

                                           way:(BTContractOrderWay)way

                                    positionID:(int64\_t)position\_id

                                         price:(NSString \*)price

                                           vol:(NSString \*)vol;

1. 发送订单

// 提交合约订单

+ (void)sendContractsOrder:(BTContractOrderModel \*)contractOrder

         contractOrderType:(BTContractOrderType)type

             assetPassword:(NSString \*)assetPassword

                   success:(void (^)(NSNumber \*))success

                   failure:(void (^)(id))failure;

// 取消合约订单

+ (void)cancelContractOrders:(NSArray <BTContractOrderModel \*>\*)contractOrders

          contractOrderType:(BTContractOrderType)type

              assetPassword:(NSString \*)assetPassword

                    success:(void (^)(NSNumber \*))success

                    failure:(void (^)(id))failure;

4. 订单没有成交的情况下会在委托深度中

// 获取用户订单记录

+ (void)getUserContractOrdersWithContractID:(int64\_t)contract\_id

                                     status:(BTContractOrderStatus)status

                                     offset:(NSInteger)offset

                                       size:(NSInteger)size

                                    success:(void (^)(NSArray <BTContractOrderModel \*>\*))success

                                    failure:(void (^)(NSError \*))failure;

传入状态BTContractOrderStatus

// 申报中

// 委托中

// 结束

1. 订单成交之后获得持仓

获得仓位有两种方式

// 根据contractID获取用户仓位

+ (void)getUserPositionWithContractID:(int64\_t)contract\_id

                               status:(BTPositionStatus)status

                               offset:(NSInteger)offset

                                 size:(NSInteger)size

                              success:(void (^)(NSArray <BTPositionModel \*>\*))success

                              failure:(void (^)(NSError \*))failure;

// 根据coinCode获取用户仓位

+ (void)getUserPositionWithcoinCode:(NSString \*)coin\_code

                         contractID:(int64\_t)contract\_id

                             status:(BTPositionStatus)status

                             offset:(NSInteger)offset

                               size:(NSInteger)size

success:(void (^)(NSArray <BTPositionModel \*>\*))success

failure:(void (^)(NSError \*))failure;

1. 合约成交记录

// 获取某一个contractID的成交记录

+ (void)getContractRecordWithContractID:(int64\_t)contract\_id

                                success:(void (^)(NSArray <BTContractTradeModel \*>\*))success

                                failure:(void (^)(NSError \*))failure;

// 获取某一个contractID的交易记录

+ (void)getContractUserRecordWithContractID:(int64\_t)contract\_id

                                    orderID:(int64\_t)order\_id

                                    success:(void (^)(NSArray <BTContractRecordModel \*>\*))success

                                    failure:(void (^)(NSError \*))failure;

// 获取用户订单记录

+ (void)getUserContractOrdersWithContractID:(int64\_t)contract\_id

                                     status:(BTContractOrderStatus)status

                                     offset:(NSInteger)offset

                                       size:(NSInteger)size

                                    success:(void (^)(NSArray <BTContractOrderModel \*>\*))success

                                    failure:(void (^)(NSError \*))failure;

3.2.5 合约资产

1. 登录之后请求获取合约资产

SLPlatformSDK

/\*\*

 \*  @brief 加载用户合约资产

 \*/

- (void)sl\_loadUserContractPerpoty;

/\*\*

 \*  @brief 该币种有没有开通账户

 \*  @param coin 币种

 \*/

- (BOOL)sl\_determineWhetherToOpenContractWithCoinCode:(NSString \*)coin;

/\*\*

 \*  @brief 获取用户单个币种的合约资产

 \*  @param coin 资产单个币种

 \*/

- (BTItemCoinModel \*)sl\_getUserFutureCoinAssetsWithCoinCode:(NSString \*)coin;

1. 获取用户资金流水

// 获得资金流水

+ (void)getCashBooksWithContractID:(int64\_t)contract\_id

                             refID:(NSArray \*)ref\_id

                            action:(NSArray \*)action

                          coinCode:(NSString \*)coin\_code

                             limit:(NSInteger)limit

                            offset:(NSInteger)offset

                             start:(NSString \*)start

                               end:(NSString \*)end

                           success:(void (^)(id result))success

                           failure:(void (^)(id ))failure;

4合约显示数据计算

1. SLFormula中有所有关于合约显示数据的计算结果

#pragma mark - 计算维持保证金率MMR

+ (NSString \*)calculateMMRWithTotalValue:(NSString \*)value

                               riskLimit:(BTContractRiskLimitModel \*)riskLimit;

#pragma mark - 计算维持保证金MM

+ (NSString \*)calculateMMWithTotalValue:(NSString \*)value

                              riskLimit:(BTContractRiskLimitModel \*)riskLimit;

#pragma mark - 计算开仓保证金率

+ (NSString \*)calculateIMRWithTotalValue:(NSString \*)value

                               riskLimit:(BTContractRiskLimitModel \*)riskLimit;

#pragma mark - 计算开仓保证金

+ (NSString \*)calculateIMWithTotalValue:(NSString \*)value

                              riskLimit:(BTContractRiskLimitModel \*)riskLimit;

#pragma mark - 计算仓位的维持保证金率

+ (NSString \*)calculatePositionMMRWithPosition:(BTPositionModel \*)position contract:(BTContract \*)contract;

#pragma mark - 计算仓位的开仓保证金率

+ (NSString \*)calculatePositionIMRWithPosition:(BTPositionModel \*)position contract:(BTContract \*)contract riskLimit:(BTContractRiskLimitModel \*)riskLimit;

#pragma mark - 计算仓位的开仓保证金

+ (NSString \*)calculatePositionIMWithPosition:(BTPositionModel \*)position riskLimit:(BTContractRiskLimitModel \*)riskLimit;

#pragma mark - 通过用户资产计算用户将资产全部开仓保证金率

+ (NSString \*)calculateOrderIMRWithAsset:(NSString \*)asset

                                 advance:(ContractOrderSize \*)advance

                                position:(BTPositionModel \*)position

                                contract:(BTContractsModel \*)contractInfo

                               riskLimit:(BTContractRiskLimitModel \*)riskLimit;

#pragma mark - 通过量和价格计算合约价值

+ (NSString \*)calculateContractValueWithVol:(NSString \*)vol

                                      price:(NSString \*)price

                                   contract:(BTContract \*)contractModel;

#pragma mark - 通过价值和量计算价格

+ (NSString \*)calculateQuotePriceWithValue:(NSString \*)value

                                       vol:(NSString \*)vol

                                  contract:(BTContract \*)contractModel;

#pragma mark - 通过价格和资产计算量

+ (NSString \*)calculateVolumeWithAsset:(NSString \*)asset

                                 price:(NSString \*)price

                                 lever:(NSString \*)lever

                               advance:(ContractOrderSize \*)advance

                              position:(BTPositionModel \*)position

                          positionType:(BTPositionType)positionType

                          contractInfo:(BTContractsModel \*)contractInfo;

#pragma mark - 根据价格和开仓方向判断是否会强平

+ (BOOL)IsLiquidateWithPositionType:(BTPositionType)positionType

positionLiquidatePrice:(NSString \*)positionLiquidatePrice

fairPrice:(NSString \*)fairPrice;

#pragma mark - 计算订单的强平价格

+ (NSString \*)calculateOrderLiquidatePriceWithOrder:(BTContractOrderModel \*)order

                                             assets:(BTItemCoinModel \*)assets

                                           contract:(BTContractsModel \*)contractModel;

#pragma mark - 计算仓位的强平价格

+ (NSString \*)calculatePositionLiquidatePrice:(BTPositionModel \*)position

                                       assets:(BTItemCoinModel \*)assets

                                 contractInfo:(BTContractsModel \*)contractInfo;

#pragma mark - 计算仓位破产价

+ (NSString \*)calculateBankruptcyPrice:(BTPositionModel \*)position

                                assets:(BTItemCoinModel \*)assets

                              contract:(BTContractsModel \*)contractModel;

#pragma mark - 计算多仓位的未实现盈亏

+ (NSString \*)calculateCloseLongProfitAmount:(NSString \*)vol

                                holdAvgPrice:(NSString \*)openPrice

                                   markPrice:(NSString \*)closePrice

                                contractSize:(NSString \*)contractSize

                                   isReverse:(BOOL)isReverse;

#pragma mark - 计算空仓位的未实现盈亏

+ (NSString \*)calculateCloseShortProfitAmount:(NSString \*)vol

                                 holdAvgPrice:(NSString \*)openPrice

                                    markPrice:(NSString \*)closePrice

                                 contractSize:(NSString \*)contractSize

                                    isReverse:(BOOL)isReverse;

#pragma mark - 计算仓位的实际杠杆

+ (NSString \*)calculatePositionLeverageWithPosition:(BTPositionModel \*)position

                                           contract:(BTContractsModel \*)contractModel;

#pragma mark - 计算订单的takeFee

+ (NSString \*)calculateOrderTakeFee:(NSString \*)orderVol

                         orderPrice:(NSString \*)orderPrice

                       takeFeeRatio:(NSString \*)takeFeeRatio

                           contract:(BTContract \*)contract;

#pragma mark - 计算订单的makeFee

+ (NSString \*)calculateOrderMakeFee:(NSString \*)orderVol

                         orderPrice:(NSString \*)orderPrice

                       makeFeeRatio:(NSString \*)makeFeeRatio

                           contract:(BTContract \*)contract;

#pragma mark - 合约单位转换，将币转成张

+ (NSString \*)coinToTicket:(NSString \*)vol

                     price:(NSString \*)price

                  contract:(BTContract \*)contract;

#pragma mark - 合约单位转换，将张转成币

+ (NSString \*)ticketToCoin:(NSString \*)vol

                     price:(NSString \*)price

                  contract:(BTContract \*)contract;

#pragma mark - 判断是否有该仓位未成交状态的平仓委托单

+ (NSArray \*)getCloseEntrustOrderWithPosition:(BTPositionModel \*)positionModel;

#pragma mark - 根据codeCoin和仓位方向获取仓位

+ (BTPositionModel \*)getUserPositionWithCoinCode:(NSString \*)codeCoin

                                      contractID:(int64\_t)contractID

                                     contractWay:(BTContractOrderWay)way;

#pragma mark - 根据合约模型和仓位方向获取仓位

+ (BTPositionModel \*)getUserPositionWithItemModel:(BTItemModel \*)itemModel

                                      contractWay:(BTContractOrderWay)way;

#pragma mark - 根据contractID获取ContractInfo

+ (BTContractsModel \*)getContractInfoWithContractID:(int64\_t)contract\_id;

#pragma mark - 通过量和价格计算合约的基础比价值

+ (NSString \*)calculateContractBasicValueWithPrice:(NSString \*)price

                                               vol:(NSString \*)vol

                                          contract:(BTContract \*)contract;

#pragma mark - 获取回应marginCode 的仓位数组

+ (NSArray \*)getPositionsWithCode:(NSString \*)coin\_code;

# 5 websocket接入

5.1 连接 websocket，见 SLSocketDataManager

/// 连接 socket, socket 连接成功后会自动订阅 市场实时价格 数据

- (void)sl\_connectSocketWithHost:(NSString \*)host;

5.2市场订阅实时价格

/\*\*

 订阅实时价格 socket

 通过监听 SLSocketDataUpdate\_Ticker\_Notification 通知获取数据

 数据格式: {@"data": BTItemModel \*}

 @param contract\_id 合约 ID, 目前只支持 0, 为 0 的时候订阅所有币种

\*/

- (void)sl\_subscribeTickerDataWithContractID:(int64\_t)contract\_id;

5.3订阅合约深度

/\*\*

 订阅深度数据 socket

 通过监听 SLSocketDataUpdate\_Depth\_Notification 通知获取数据

 数据格式: {@"buys": NSArray <BTOrderModel \*> \*, @"sells": NSArray <BTOrderModel \*> \*}, buys 和 sells 一般不同时存在

 @param contract\_id 合约 ID

 \*/

- (void)sl\_subscribeDepthDataWithContractID:(int64\_t)contract\_id;

5.4订阅最新成交

/\*\*

 订阅最新成交 socket

 通过监听 SLSocketDataUpdate\_Trade\_Notification 通知获取数据

 数据格式:

 @param contract\_id 合约 ID

 \*/

- (void)sl\_subscribeTradeDataWithContractID:(int64\_t)contract\_id

5.5订阅K 线图数据

/// 订阅 k 线图数据 socket

/// 通过监听 SLSocketDataUpdate\_QuoteBin\_Notification 通知获取数据

/// 数据格式: {@"data": NSArray <BTItemModel \*> \*, @"kLineDataType": @(SLStockLineDataType)}

/// @param contract\_id   合约 ID

/// @param kLineDataType  k 线图类型

- (void)sl\_subscribeQuoteBinDataWithContractID:(int64\_t)contract\_id stockLineDataType:(SLStockLineDataType)kLineDataType;

5.6订阅私有信息

/\*\*

 \* 订阅私有信息(订单信息, 仓位信息, 合约资产, 现货资产)

 \*/

- (void)sl\_subscribeUnicastData;

注：sdk初始化流程

未登录状态首页

1. SLSDK初始化
2. 请求合约市场
3. 监听市场ticker

交易页面

1. 监听合约深度同时监听市场ticker
2. 切换合约的时候切换监听的合约深度id
3. 登录之后监听私有信息（仓位订单还有资产变化）

市场详情页

1. 监听市场ticker
2. 监听对应合约成交记录。合约深度以及K线退出是取消订阅

资产页面

1. 监听私有信息（仓位资产变化）
2. 监听市场ticker改变（可能会改变仓位的为实现盈亏）