dFuture API说明

交易接口

1. 增加保证金: deposit
2. 提取保证金: withdraw
3. 止盈止损单: closePositionFromOrderBook
4. 线下价格开仓: openPositionWithPrice
5. 线下价格平仓: closePositionWithPrice
6. 链上接口开仓(带截止日期): openPositionWithDeadLine
7. 链上接口平仓(带截止日期): closePositionWithDeadLine
8. 链上接口开仓: openPosition
9. 链上接口平仓: closePosition
10. 强平接口: forceClosePosition
11. 提取手续费和利息费奖励: claimAllUSDT

流动性接口

1. 增加流动性: depositToPool
2. 减少流动性: withdrawFromPool
3. 触发给LP分配奖励: distributeFeesOfLP

查询接口

1. 查询手续费利息费奖励信息: queryIOUFeesAndInterests
2. 查询持仓信息: queryPosition
3. 查询开仓/平仓的手续费和费率: queryPositionFeeAndRatio
4. 查询各奖励池的收益情况
5. 查询交易者持仓信息: queryHolderInfo
6. 查询Fomo信息: queryFomoInfo
7. 查询已经确定仓位的利息费: queryInterestRatio
8. 查询交易者所持仓位下一期的利息(预估): queryPendingInterest
9. 查询交易者享受的额外手续费折扣率: queryExtraDiscount
10. 查询链上交易接口是否可用: queryIfEmergency
11. 查询线下价格开/平仓接口手续费参数

交易接口

1. 增加保证金: deposit

- 方法原型: `function deposit(bytes32 _symbol, uint256 _usdtAmount) whenNotClearing`
- 参数说明:
  - _symbol: 交易标的, 如'btc', 'eth', 'uni'等, bytes32编码
  - _usdtAmount: 增加的保证金额度, 18位精度.
- 事件:
  `event Deposit(address indexed who, bytes32 symbol, uint256 amount);`

增加保证金成功, 将发出 `Deposit` 事件.

2. 提取保证金: withdraw

- 方法原型: `function withdraw(bytes32 _symbol, uint256 _usdtAmount) whenNotPaused`
- 参数说明:
  - _symbol: 交易标的, 如'btc', 'eth', 'uni'等, bytes32编码
  - _usdtAmount: 提取的保证金额度, 18位精度.
- 事件:
  `event Withdraw(address indexed who, bytes32 symbol, uint256 amount);`
- 限制:
  提取保证金后, 剩余仓位的保证金不得低于平台最低保证金率要求.

3. 止盈止损单: closePositionFromOrderBook

- 方法原型:

```
function closePositionFromOrderBook(
        address maker,
        bytes32 _symbol,
```

```
        uint256 _amount,
        address relayer,
        uint256 reward
    ) onlyOrderBook
```

- 参数说明:
  - maker: 止盈止损单下单账号
  - _symbol: 标的
  - _amount: 平仓数量, 表示手数
  - relayer: 交易中继人账号
  - reward: 交易中继人奖励
- 限制:
  只能由订单本合约发起调用.

4. 线下价格开仓: openPositionWithPrice

- 方法原型:
  `function openPositionWithPrice(Orders.OpenOrder memory order, Orders.OfflinePrice memory prices) whenNotPaused onlyRelayer`
- 参数说明:
  - order: 由用户签名的订单信息.

```
struct OpenOrder {
        bytes32 symbol;
        uint256 amount;
        int8    direction;
        uint256 acceptablePrice;
        uint256 approvedUsdt;
        address parent;
        bool    withDiscount;
        uint256 deadline;
        address maker;
        uint8   gasLevel;

        uint8 v;
        bytes32 r;
        bytes32 s;
    }
```

- prices: 当前价格信息

```
struct OfflinePrice {
        uint256 price;
        uint    timestamp;
    }
```

- 限制:
  仅由指定的中继账号发起调用.

5. 线下价格平仓: closePositionWithPrice

- 方法原型:
  `function closePositionWithPrice(Orders.OpenOrder memory order, Orders.OfflinePrice memory prices) whenNotPaused onlyRelayer`
- 参数说明:
  - order: 由用户签名的订单信息.

```
    struct CloseOrder {
        bytes32 symbol;
        uint256 amount;
        uint256 acceptablePrice;
        uint256 deadline;
        address maker;
        uint8   gasLevel;

        uint8 v;
        bytes32 r;
        bytes32 s;
    }
```

  - prices: 当前价格信息
- 限制:

仅由指定的中继账号发起调用.

6. 链上接口开仓(带截止日期): openPositionWithDeadLine

- 方法原型:

```
function openPositionWithDeadLine(
```

```
        bytes32 _symbol,
        uint256 _amount,
        int8 _direction,
        uint256 _acceptablePrice,
        uint256 _approvedUsdt,
        address parent,
        bool withDiscount,
        uint deadline
    ) external whenNotPaused onlyEOA onlyEmergency
```

- 参数说明:
  - parent: 上级关系人账号
  - withDiscount: 是否享受手续费折扣
  - deadline: 本交易可接受成交截止时间. epoch秒数.
    其余参数意义同前.
- 限制:
  - 仅限于非合约账号.
  - 仅限于 `线下价格开仓` 服务不可用.

7. 链上接口平仓(带截止日期): closePositionWithDeadLine

- 方法原型:

```
function closePositionWithDeadLine(
        bytes32 _symbol,
        uint256 _amount,
        uint256 _acceptablePrice,
        uint deadline
    ) external whenNotPaused onlyEOA onlyEmergency
```

参数和限制与 `openPositionWithDeadLine` 相同.

8. 链上接口开仓: openPosition

- 方法原型:

```
function openPosition(
        bytes32 _symbol,
        uint256 _amount,
        int8    _direction,
```

```
        uint256 _acceptablePrice,
        uint256 _approvedUsdt,
        address parent,
        bool withDiscount
    ) external whenNotPaused onlyEOA onlyEmergency
```

除了无交易成交截止时间要求外, 与 `openPositionWithDeadLine` 方法相同.

9. 链上接口平仓: closePosition

- 方法原型:

```
function closePosition(
        bytes32 _symbol,
        uint256 _amount,
        uint256 _acceptablePrice
    ) external whenNotPaused onlyEOA onlyEmergency
```

除了无交易成交截止时间要求外, 与 `closePositionWithDeadLine` 方法相同.

10. 强平接口: forceClosePosition

- 方法原型:

```
function forceClosePosition(
        address _holder,
        bytes32 _symbol,
        address[] calldata compAddr,
        bytes32[] calldata compSymbol
    )
        external
        whenNotPaused
```

- 参数说明:
  - _holder: 需要被强平仓位的持仓人账号.
  - _symbol: 仓位标的
  - compAddr: 如果被强平仓位已经穿仓, 补偿穿仓损失的账号列表.
  - compSymbol: 如果被强平仓位已经穿仓, 补偿穿仓损失的标的列表.
    `compAddr` 和 `compSymbol` 的长度需要匹配.

- 限制:
  仅保证金率低于平台要求的仓位可以被强平.

11. 提取手续费和利息费奖励: claimAllUSDT

- 方法原型: `function claimAllUSDT() external whenNotPaused`
- 参数说明: 无
- 限制:
  手续费池累积收益大于手续费奖励时, 可提取手续费奖励.
  利息费池中累积收益大于利息费奖励时, 可提取利息费奖励.

流动性接口

1. 增加流动性: depositToPool

- 方法原型:

```
function depositToPool(uint256 _usdtAmount)
        external
        whenNotClearing
        onlyEOA
```

- 参数说明:
  - _usdtAmount: 提供的流动性数量, 18位精度.
- 限制:
  仅限于非合约账号.

2. 减少流动性: withdrawFromPool

- 方法原型:

```
function withdrawFromPool(uint256 _lpTokenAmount)
        external
        whenNotPaused
        onlyEOA
```

- 参数说明
  - _lpTokenAmount: 提取的LpToken的数额
- 限制:

仅限于非合约账号.

3. 触发给LP分配奖励: distributeFeesOfLP

- 方法原型:

```
function distributeFeesOfLP() external whenNotPaused
```

- 限制:
  - 当手续费池净累积收益(扣除交易者奖励) 大于一定额度(1000USDT)时, 可以分配手续费.
  - 当利息费池净累积收益(扣除交易者奖励) 大于一定额度(1000USDT)时, 可以分配利息费.

查询接口

1. 查询手续费利息费奖励信息: queryIOUFeesAndInterests

- 方法原型:

```
function queryIOUFeesAndInterests(address _holder)
        external
        view
        returns(uint256 oweFees, int256 oweInterests, uint256 iouT
otalFees, uint256 iouTotalInterest)
```

- 输入:
  - _holder: 要查询的账号
- 输出:
  - oweFees: 平台总共需要支付的手续费奖励.
  - oweInterests: 平台总共需要支付的利息费奖励.
  - iouTotalFees: 需要支付 `_holder` 的手续费.
  - iouTotalInterest: 需要支付 `_holder` 的利息费.

2. 查询持仓信息: queryPosition

- 方法原型:

```
function queryPosition(address _holder, bytes32 symbol)
        external
        view
        returns (FPTypes.Position memory)
```

- 输入:

    - _holder: 查询账号
    - symbol: 查询标的
- 输出: 持仓信息

```
struct Position {
        uint104   amount;       // 持仓数量
        uint104   price;        // 开仓均价
        uint40    modifyBlock; // 最后一次操作时的块高
        int8      direction;    // 仓位方向: 1 多仓, -1 空仓
    }
```

3. 查询开仓/平仓的手续费和费率: queryPositionFeeAndRatio

查询如果开仓/平仓, 需要被收取/被奖励的手续费和费率.

- 方法原型:

```
function queryPositionFeeAndRatio(
        bytes32 _symbol,
        uint256 _amount,
        int8 _direction,
        bool _isForOpen
    )
        external
        view
        returns (int256 fee, int256 ratio, int256 oldratio)
```

- 输入:
    - _symbol: 标的
    - _amount: 数量
    - _direction: 仓位方向
    - _isForOpen: 是否开仓. `true` 表示开仓, `false` 表示平仓
- 输出:
    - fee: 手续费, 如果为负值, 表示被奖励; 如果为正值, 表示被收取.
    - ratio: 抵押DFT情况的费率.
    - oldratio: 不抵押DFT情况下的费率.

4. 查询各奖励池的收益情况

- 方法原型:

```
function queryBalanceOfFees()
        external
        view
        returns(FPTypes.PositionFeeDeposit memory)
```

- 输入: 无
- 输出:

```
struct PositionFeeDeposit {
        uint256 lpFees;     // 手续费池现累积收益
        uint256 lpInterest; // 利息费池现累积收益

        uint128 reserved;   // 保留池
        uint128 reserved1;  // 保留池1
    }
```

5. 查询交易者持仓信息: queryHolderInfo

- 方法原型:

```
function queryHolderInfo(address _holder, bytes32 _symbol)
        external
        view
        returns (
            uint256 balance,
            uint256 marginRatio,
            int256 profit,
            uint256 value
        )
```

- 输入:
  - _holder: 交易者账号
  - _symbol: 持仓标的
- 输出:
  - balance: 本标的保证金数额

- marginRatio: 保证金率
- profit: 本仓位的盈亏
- value: 仓位

6. 查询Fomo信息: queryFomoInfo

- 方法原型:

```
function queryFomoInfo()
        external
        view
        returns (
            address trader,
            uint    blocknum
        )
```

- 输入: 无
- 输出:
  - trader: 上一次交易者账号
  - blocknum: 上一次交易时的块高

7. 查询已经确定仓位的利息费: queryInterestRatio

- 方法原型:

```
function queryInterestRatio(bytes32 symbol, int128 index)
        external
        view
        returns(int256 longRatio, int256 shortRatio, uint128 curre
ntIndex, uint128 updatedIndex)
```

- 输入:
  - symbol: 标的
  - index: 期数. index = 0, 表示最近已经确定的利息率. index = -1 表示已经确定的上一期的利息率. 依次类推.
- 输出:
  - longRatio: 指定期数的多仓利息率.
  - shortRatio: 指定期数的空仓利息率.
  - currentIndex: 当前所在期数.
  - updatedIndex: 已经确定利息率的最近的期数.

8. 查询交易者所持仓位下一期的利息(预估): queryPendingInterest

- 方法原型:

```
function queryPendingInterest(address holder, bytes32 symbol)
        external
        view
        returns (int256)
```

- 输入:
  - holder: 交易者账号
  - symbol: 标的
- 输出:
  - 利息

9. 查询交易者享受的额外手续费折扣率: queryExtraDiscount

- 方法原型:

```
function queryExtraDiscount(address trader) external view returns(
uint8)
```

- 输入:
  - trader: 交易者
- 输出:
  折扣率, 百分比表示. eg: 返回85表示 85%折扣, 依此类推.

10. 查询链上交易接口是否可用: queryIfEmergency

- 方法原型:

```
function queryIfEmergency() external view returns(bool)
```

- 输入: 无
- 输出:
  bool值, `true` 表示链上交易接口可用, `false` 表示链上交易接口不可用.

11. 查询线下价格开/平仓接口手续费参数

- 方法原型:

```
function queryOfflineServiceFeeParameters() external view returns(
uint256 gasLimit, uint256 exchangeRate)
```

- 输入: 无
- 输出:
  - gasLimit: 需要收取的gasLimit.
  - exchangeRate: gasPrice与运营平台本币的兑换比例. 如: 在Heco上运营, exchangeRate表示HT/USDT的兑换率, 在BSC上表示BNB/USDT的兑换率.

12. 查询交易者线下价格开/平仓的交易Nonce: queryNonce

- 方法原型:

```
function queryNonce(address trader) external view returns (uint64)
```

- 输入:
  - trader: 交易者账号
- 输出:
  uint64, 表示已经成交的交易数量.

LiqiudPool信息查询接口

1. Pool中总体信息: getInfo

- 方法原型:

```
function getInfo()
        public
        view
        returns (
            uint256 balance,
            uint256 marginRatio,
            int256 profit,
            uint256 value,
            uint256 longPositionValue,
            uint256 shortPositionValue
        )
```

- 输入: 无
- 输出:
  - balance: 流动性总额
  - marginRatio: Pool的保证金率
  - profit: Pool的盈亏
  - value: Pool的总仓位
  - longPositionValue: Pool中总的多仓仓位.(对应交易者的总共空仓仓位)
  - shortPositionValue: Pool中总的空仓仓位.

2. Pool指定标的的仓位:

- 方法原型:

```
function getPositionOf(bytes32 symbol)
        public
        view
        returns(FPTypes.Position[] memory)
```

- 输入:
  - symbol: 标的
- 输出:
  - [0]: 指定标的在Pool中的总多仓信息
  - [1]: 指定标的在Pool中的总空仓信息

3. Pool中多/空仓偏差及偏差率

- 方法原型:

```
function getPositionOffsetAndRatio() external view returns(int256
offset, uint256 ratio)
```

- 输入: 无
- 输出:
  - offset: 多/空仓偏差, 总多仓仓位 - 总空仓仓位.
  - ratio: 多比空的偏差率 = offset / (Pool流动性总额 + Pool的盈亏)