Contract Interface Documentation

1. Pool Contract (`contracts/pool/pool.tact`)

1.1 Liquidity Operations

Add Liquidity (Increase LP Position)

- **Purpose:** Users add Jetton to the pool and receive TLP shares.
- **Parameter Description:**

Name	Type	Description	Example
amount	uint	Amount of Jetton	100000
executionFee	coins	TON for order execution	0.1

- **Payload Structure:**
  - 1 bit: 1 (operation flag)
  - ref: cell
    - 8 bits: 1 (operation code)
    - coins: executionFee
- **TypeScript Example:**

```
beginCell()
  .storeUint(1, 1)
  .storeRef(
    beginCell()
      .storeUint(1, 8) // OP_CREATE_INCREASE_LP_POSITION_ORDER = 1
      .storeCoins(toNano(executionFee))
      .endCell()
  ).endCell().asSlice()
```

- **Key Contract Interactions:**
  - `JettonTransfer` (add liquidity payload)
  - `ExecuteLiquidityOrder`
- **Related Events:**
  - `LiquidityOrderCreatedEvent`

- `LiquidityPoolChangedEvent`

Remove Liquidity (Decrease LP Position)

- **Purpose:** Users redeem TLP to receive Jetton (USDT).
- **Parameter Description:**

Name	Type	Description	Example
amount	uint	Amount of TLP	100
executionFee	coins	TON for order execution	0.1

- **Payload Structure:**
  - 1 bit: 1 (operation flag)
  - ref: cell
    - coins: executionFee
- **TypeScript Example:**

```
beginCell()
  .storeUint(1, 1)
  .storeRef(
    beginCell()
      .storeCoins(toNano(executionFee))
      .endCell()
  ).endCell().asSlice()
```

- **Key Contract Interactions:**
  - `JettonTransfer` (remove liquidity payload)
  - `ExecuteLiquidityOrder`
- **Related Events:**
  - `LiquidityOrderCreatedEvent`
  - `LiquidityPoolChangedEvent`

ExecuteLiquidityOrder

- **Purpose:** Called by the executor to process pending liquidity orders.
- **Parameter Description:**

Name	Type	Description
orderId	uint64	Liquidity order ID

Name	Type	Description
trxId	uint64	Transaction ID
executionFeeReceiver	Address?	Address to receive fee
prices	map	Price data
lpFundingFeeGrowth	coins	Funding fee growth
rolloverFeeGrowth	coins	Rollover fee growth

- **TypeScript Example:**

```
await pool.send(
  provider.sender(),
  { value: toNano('0.5') },
  {
    $$type: 'ExecuteLiquidityOrder',
    orderId: orderId,
    trxId: 2n,
    executionFeeReceiver: provider.sender().address!!,
    prices: prices,
    lpFundingFeeGrowth: toJettonUnits(10),
    rolloverFeeGrowth: toJettonUnits(10),
  }
);
```

- **Related Events:**
  - `LiquidityPoolChangedEvent`

1.2 Perpetual Trading Operations

Open Perpetual Position (Increase Perp Position)

- **Purpose:** Users open a perpetual contract position.
- **Parameter Description:**

Name	Type	Description	Example
margin	uint	Margin	1000
size	uint	Position size	10000

Name	Type	Description	Example
isLong	bool	Direction (long/short)	true
triggerPrice	uint128	Trigger price	123456
executionFee	coins	TON for order execution	0.1

- **Payload Structure:**
  - 1 bit: 1 (operation flag)
  - ref: cell
    - 8 bits: 2 (operation code)
    - coins: executionFee
    - 1 bit: isMarket
    - 16 bits: tokenId
    - 1 bit: isLong
    - coins: size
    - 128 bits: triggerPrice
    - 32 bits: requestTime
    - ref: cell (TP/SL)
      - coins: tpSize
      - 128 bits: tpPrice
      - coins: slSize
      - 128 bits: slPrice
- **TypeScript Example:**

```
beginCell()
  .storeUint(1, 1)
  .storeRef(
    beginCell()
      .storeUint(2, 8)
      .storeCoins(toNano(executionFee))
      .storeBit(true) // isMarket
      .storeUint(tokenId, 16)
      .storeBit(isLong)
      .storeCoins(toUnits(size, MOCK_DECIMAL))
      .storeUint(toUnits(triggerPrice, PRICE_DECIMAL), 128)
      .storeUint(now(), 32)
      .storeRef(
        beginCell()
          .storeCoins(toUnits(tpSize, MOCK_DECIMAL))
          .storeUint(toUnits(tpPrice, PRICE_DECIMAL), 128)
          .storeCoins(toUnits(slSize, MOCK_DECIMAL))
          .storeUint(toUnits(slPrice, PRICE_DECIMAL), 128)
      )
      .endCell()
  ).endCell().asSlice()
```

- **Key Contract Interactions:**
  - `JettonTransfer` (open position payload)
  - `ExecutePerpOrder`
- **Related Events:**
  - `PerpOrderCreatedEvent`
  - `PerpPositionIncreasedEvent`

Close/Decrease Perpetual Position

- **Purpose:** Users close or decrease a position.
- **Parameter Description:**

Name	Type	Description	Example
tokenId	uint16	Trading pair ID	1
margin	uint	Margin	1000
size	uint	Amount to decrease	5000
triggerPrice	uint128	Trigger price	123456

Name	Type	Description	Example
executionFee	coins	TON for order execution	0.1

- **Key Contract Interactions:**
  - `CreateDecreasePerpOrder`
  - `ExecutePerpOrder`
- **Related Events:**
  - `PerpOrderCreatedEvent`
  - `PerpPositionDecreasedEvent`

LiquidatePerpPosition

- **Purpose:** Liquidate positions with low margin ratio.
- **Parameter Description:**

Name	Type	Description
positionId	uint64	Position ID
executor	Address	Executor address

- **Related Events:**
  - `PerpPositionDecreasedEvent`

ADLPerpPosition

- **Purpose:** Auto-deleveraging.
- **Parameter Description:**

Name	Type	Description
positionId	uint64	Position ID
executor	Address	Executor address

- **Related Events:**
  - `PerpPositionDecreasedEvent`

Cancel Perpetual Order

- **Purpose:** Cancel an existing perpetual order before it is executed.
- **Parameter Description:**

Name	Type	Description	Example
orderId	uint64	The ID of the perpetual order	123
trxId	uint64	Transaction identifier	1
executionFeeReceiver	Address?	Address to receive execution fee	0:abc...

- **TypeScript Example:**

```
// Assume provider and pool are available
const orderId = 123n;
const trxId = 1n;
await pool.send(
  provider.sender(),
  { value: toNano('0.2') },
  {
    $$type: 'CancelPerpOrder',
    orderId: orderId,
    trxId: trxId,
    executionFeeReceiver: provider.sender().address!!
  }
);
```

- **Key Contract Interactions:**
  - `CancelPerpOrder`
- **Related Events:**
  - `PerpOrderCancelledEvent`

Create Standalone Take-Profit/Stop-Loss Order

- **Purpose:** Create a standalone take-profit/stop-loss (TP/SL) order for an existing position (not attached at open).
- **Parameter Description:**

Name	Type	Description	Example
tokenId	uint16	Trading pair ID	1
isLong	bool	Position direction	true
tpSize	coins	Take-profit size	0.01
tpPrice	uint128	Take-profit price	61000

Name	Type	Description	Example
slSize	coins	Stop-loss size	0.01
slPrice	uint128	Stop-loss price	59000
executionFee	coins	TON for order execution	0.3
trxId	uint64	Transaction identifier	1
requestTime	uint32	Request timestamp	1680000000

- **TypeScript Example:**

```
// Assume provider and pool are available
const executionFee = 0.3;
const tokenId = 1;
const isLong = true;
const tpSize = toUnits(0.01, MOCK_DECIMAL);
const tpPrice = toUnits(61000, PRICE_DECIMAL);
const slSize = toUnits(0.01, MOCK_DECIMAL);
const slPrice = toUnits(59000, PRICE_DECIMAL);
const trxId = 1n;
const requestTime = BigInt(Math.floor(Date.now() / 1000));
await pool.send(
  provider.sender(),
  { value: toNano(executionFee + 0.2) },
  {
    $$type: 'CreateTpSlPerpOrder',
    executionFee: toNano(executionFee),
    tokenId: tokenId,
    isLong: isLong,
    tpSize: tpSize,
    tpPrice: tpPrice,
    slSize: slSize,
    slPrice: slPrice,
    trxId: trxId,
    requestTime: requestTime,
  }
);
```

- **Key Contract Interactions:**
  - `CreateTpSlPerpOrder`
- **Related Events:**

- `PerpOrderCreatedEvent`
- `PerpPositionDecreasedEvent` (when TP/SL is executed)

2. Constants & Parameter Description

- `OP_CREATE_INCREASE_LP_POSITION_ORDER = 1`
- `OP_CREATE_INCREASE_PERP_POSITION_ORDER = 2`