SunSwap 2.0 Interface Documentation

-- Sun Team

v 1.0.4

Table of Contents

I. Backend API

1.1 Access all trading pairs

GET: https://openapi.sun.io/v2/allpairs?ver=3

Parameters:

page_size : int, size of each page, max 500

page_num: int, number of the page, starting from 0

token_address: optional; return only the data containing the specific token address

orderBy: optional; return the data ordered by the specific item (price, quote_volume or base_volume)

desc: optional; true = decreasing order; false = increasing order

Return format:

```
{
        "data": [{
"TNUC9Qb1rRpS5CbWLmNMxXBjyFoydXjWFR_TR7NHqjeKQxGTCi8q8ZY4pL8otSzgjLj6tf":
//key: ids of base_token and quote_token
 {
                "quote_id": "TNUC9Qb1rRpS5CbWLmNMxXBjyFoydXjWFR",
                "quote_decimal": "6", //precision of quote_token
                "quote_name": "Wrapped TRX", //name of quote_token
                "quote_symbol": "WTRX", //symbol of quote_token
                "base_id": "TR7NHqjeKQxGTCi8q8ZY4pL8otSzgjLj6t", //address of base_token
                "base_decimal": "6", //precision of base_token
                "base_name": "Tether USD", //name of base_token
                "base_symbol": "USDT", //symbol of base_token
                "price": "0.938196997790940827", //price of quote_token, calculated in base_token
                "quote_volume": "0", //total amount of quote_token traded in the last 24 hours
(minimum unit)
                "base_volume": "0" //total amount of base_token traded in the last 24 hours
(minimum unit)
                }
        }],
        "err_msg": "",
        "err_no": 0,
        "total_num": 143 //total number of entries
}
```

II. Smart Contract Interface

2.1 Smart contract address

Name	Address	Note
factory contract	TKWJdrQkqHisa1X8HUdHEfREvTzw4pMAaY	Factory contract is designed to create trading pairs and manage lists of trading pairs.
router contract	TKzxdSv2FZKQrEqkKVgp5DcwEXBEKMg2Ax	
pair contract	TFGDbUyP8xez44C76fin3bn3Ss6jugoUwJ	Each trading pair has a pair contract and it is the contract address for the USDT/TRX pair.

2.2 List of contract interfaces

2.2.1 TRC20 token

```
interface ITRC20 {
  function transfer(address to, uint256 value) external returns (bool);
  function approve(address spender, uint256 value) external returns (bool);
  function transferFrom(address from, address to, uint256 value) external returns (bool);
  function totalSupply() external view returns (uint256);
  function balanceOf(address who) external view returns (uint256);
  function allowance(address owner, address spender) external view returns (uint256);
  event Transfer(address indexed from, address indexed to, uint256 value);
  event Approval(address indexed owner, address indexed spender, uint256 value);
}
```

2.2.2 Factory

```
interface ISunswapV2Factory {
    event PairCreated(address indexed token0, address indexed token1, address pair, uint);

    function feeTo() external view returns (address);
    function feeToSetter() external view returns (address);

    function getPair(address tokenA, address tokenB) external view returns (address pair);
    function allPairs(uint) external view returns (address pair);
    function allPairsLength() external view returns (uint);

    function createPair(address tokenA, address tokenB) external returns (address pair);

    function setFeeTo(address) external;
    function setFeeToSetter(address) external;
}
```

2.2.3 ISunswapV2Router

```
interface ISunswapV2Router01 {
    function factory() external pure returns (address);
    function WETH() external pure returns (address);

    function addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external returns (uint amountA, uint amountB, uint liquidity);
    function addLiquidityETH(
        address token,
        uint amountTokenDesired,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external payable returns (uint amountToken, uint amountETH, uint liquidity);
    function removeLiquidity(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external returns (uint amountA, uint amountB);
    function removeLiquidityETH(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external returns (uint amountToken, uint amountETH);
    function removeLiquidityWithPermit(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountA, uint amountB);
    function removeLiquidityETHWithPermit(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountToken, uint amountETH);
    function swapExactTokensForTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external returns (uint[] memory amounts);
    function swapTokensForExactTokens(
        uint amountOut,
        uint amountInMax,
        address[] calldata path,
        address to,
        uint deadline
    ) external returns (uint[] memory amounts);
    function swapExactETHForTokens(uint amountOutMin, address[] calldata path, address to, uint
deadline)
        external
        payable
        returns (uint[] memory amounts);
    function swapTokensForExactETH(uint amountOut, uint amountInMax, address[] calldata path,
address to, uint deadline)
        external
        returns (uint[] memory amounts);
    function swapExactTokensForETH(uint amountIn, uint amountOutMin, address[] calldata path,
address to, uint deadline)
        external
        returns (uint[] memory amounts);
    function swapETHForExactTokens(uint amountOut, address[] calldata path, address to, uint
deadline)
        external
        payable
        returns (uint[] memory amounts);

    function quote(uint amountA, uint reserveA, uint reserveB) external pure returns (uint amountB);
    function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) external pure returns (uint
amountOut);
    function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) external pure returns (uint
amountIn);
    function getAmountsOut(uint amountIn, address[] calldata path) external view returns (uint[]
memory amounts);
    function getAmountsIn(uint amountOut, address[] calldata path) external view returns (uint[]
memory amounts);
}
```

2.3 Description of contract interfaces

2.3.1 Factory

2.3.1.1 Query interface

1. getPair

function getPair(address tokenA, address tokenB) external view returns (address pair);

Function description: Use tokenA and tokenB to acquire the corresponding pair address. Return the zero address (0x0000000000000000000000000000000000000000) if the pair address is not created.

Parameter description:

Paramete r	Type	Description
tokenA	address	TRC20 token address
tokenB	address	TRC20 token address

Returns:

address	Trading pair address

2. allPairs

function allPairs(uint) external view returns (address pair);

Function description: Return the address of the trading pair N (0-indexed) created by the factory contract. If the corresponding index has not been created, return the zero address (0x0000000000000000000000000000000000000000000000000000000000000000).

Parameter description:

Parameter	Type	Description
index	uint256	TRC20 token address

Returns:

address	Trading pair address

3. allPairsLength

function allPairsLength() external view returns (uint);

Function description: Return the number of trading pairs created via the factory contract.

Returns:

uint	Total number of trading pairs

2.3.1.2 Modification interface

1. createPair

function createPair(address tokenA, address tokenB) external returns (address pair);

Function description: Create a pair address for tokenA and tokenB when their corresponding trading pair does not exist.

Parameter description:

Parameter	Type	Description
tokenA	address	TRC20 token address
tokenB	address	TRC20 token address

Returns:

address	Trading pair address

2.3.1.3 Contract event

1. PairCreated

event PairCreated(address indexed token0, address indexed token1, address pair, uint);

Function description: The interface sends an event when creating trading pair with createPair.

Parameter description:

Parameter	Type	Description
token0	address	TRC20 token address
token1	address	TRC20 token address

pair	address	TRC20 token's corresponding exchange address in JustSwap
index	uint	The final uint log value for the first trading pair created is 1, the value for the second pair is 2, and so on.

2.3.2 Router

2.3.2.1 Query interface

1. factory

function factory() external pure returns (address);

Function description: Return the factory contract address.

Returns:

address	factory contract address

2. WETH

function WETH() external pure returns (address);

Function description: Return the WTRX contract address.

Returns:

address	WTRX contract address

2.3.2.2 Modification interface

1. addLiquidity

```
 function addLiquidity(
  address tokenA,
  address tokenB,
  uint amountADesired,
  uint amountBDesired,
  uint amountAMin,
  uint amountBMin,
  address to,
  uint deadline
) external returns (uint amountA, uint amountB, uint liquidity);
```

Function description:

Add liquidity to the TRC-20 ⇄ TRC-20 pool, and then mint liquidity tokens as markers.

Approve before adding liquidity.

Parameter description:

Parameter	Type	Description
tokenA	address	TRC20 token address
tokenB	address	TRC20 token address
amountADesired	uint256	Add this amount as the liquidity of tokenA if the price of B/A is <= amountBDesired/amountADesired (i.e. A depreciates).
amountBDesired	uint256	Add this amount as the liquidity of tokenB if the price of A/B is <= amountADesired/amountBDesired (i.e. B depreciates).
amountAMin	uint256	This limit value calculated from the slippage must be lower than amountADesired.

amountBMin	uint256	This limit value calculated from the slippage must be lower than amountBDesired.
to	address	Address receiving liquidity token
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

Returns:

amountA	uint256	Amount of tokenA sent to the pool
amountB	uint256	Amount of tokenB sent to the pool
liquidity	uint256	Additional amount of liquidity token issued to the caller

2. addLiquidityETH

```
function addLiquidityETH(
  address token,
  uint amountTokenDesired,
  uint amountTokenMin,
  uint amountETHMin,
  address to,
  uint deadline
) external payable returns (uint amountToken, uint amountETH, uint liquidity);
```

Function description:

Add liquidity to the TRC-20 ⇄ WTRX pool with TRX, and then mint liquidity tokens as markers.

Approve before adding liquidity.

Parameter description:

Parameter	**Type**	**Description**
token	address	TRC20 token address

amountTokenDesired	uint256	Add this amount as the liquidity of token if the price of WTRX/token is <= msg.value/amountTokenDesired (i.e. token depreciates)
msg.value (amountETHDesired)	uint256	Add this amount as the liquidity of WTRX if the price of token/WTRX is<= amountTokenDesired/msg.value (i.e. WTRX depreciates)
amountTokenMin	uint256	This limit value calculated from the slippage must be lower than amountTokenDesired
amountETHMin	uint256	This limit value calculated from the slippage must be lower than msg.value
to	address	Address receiving liquidity token
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

Returns:

amountToken	uint256	Amount of token sent to the pool
amountETH	uint256	Amount of TRX converted to WTRX and sent to the pool
liquidity	uint256	Additional amount of liquidity token issued to the caller

3. removeLiquidity

```
function removeLiquidity(
  address tokenA,
  address tokenB,
  uint liquidity,
  uint amountAMin,
  uint amountBMin,
  address to,
  uint deadline
) external returns (uint amountA, uint amountB);
```

Function description:

Remove liquidity from the TRC-20 ⇄ TRC-20 pool. Approval is needed before the removal of liquidity.

Parameter description:

Parameter	**Type**	**Description**
tokenA	address	TRC20 token address
tokenB	address	TRC20 token address
liquidity	uint256	Amount of the liquidity token to be removed
amountAMin	uint256	Minimum amount of tokenA to be received; calculated from the slippage
amountBMin	uint256	Minimum amount of tokenB to be received; calculated from the slippage
to	address	Address receiving tokenA/tokenB
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

Returns:

amountA	uint256	Amount of tokenA received
amountB	uint256	Amount of tokenB received

4. removeLiquidityETH

```
function removeLiquidityETH(
  address token,
  uint liquidity,
  uint amountTokenMin,
  uint amountETHMin,
  address to,
  uint deadline
) external returns (uint amountToken, uint amountETH);
```

Function description:

Remove liquidity from the TRC-20 ⇄ WTRX pool and convert WTRX into TRX. Approval is needed before the removal of liquidity.

Parameter description:

Parameter	Type	Description
token	address	TRC20 token address
liquidity	uint256	Amount of the liquidity token to be removed
amountTokenMin	uint256	Minimum amount of the token to be received; calculated from the slippage
amountETHMin	uint256	Minimum amount of TRX to be received; calculated from the slippage
to	address	Address receiving token/TRX
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

Returns:

amountToken	uint256	Amount of token received
amountETH	uint256	Amount of TRX received

5. removeLiquidityWithPermit

```
function removeLiquidityWithPermit(
  address tokenA,
  address tokenB,
  uint liquidity,
  uint amountAMin,
  uint amountBMin,
  address to,
  uint deadline,
  bool approveMax, uint8 v, bytes32 r, bytes32 s
) external returns (uint amountA, uint amountB);
```

Function description:

Remove liquidity from the TRC-20 ⇄ TRC-20 pool. Approval is not needed when removing liquidity due to the existence of permit.

Parameter description:

Parameter	**Type**	**Description**
tokenA	address	TRC20 token address
tokenB	address	TRC20 token address
liquidity	uint256	Amount of the liquidity token to be removed
amountAMin	uint256	Minimum amount of tokenA to be received; calculated from the slippage
amountBMin	uint256	Minimum amount of tokenB to be received; calculated from the slippage
to	address	Address receiving tokenA/tokenB
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit
approveMax	bool	Whether the amount to be approved by the signature is uint(-1) or liquidity
v	uint8	v component in permit signature
r	bytes32	r component in permit signature
s	bytes32	s component in permit signature

Returns:

amountA	uint256	Amount of tokenA received

amountB	uint256	Amount of tokenB received

6. removeLiquidityETHWithPermit

```
function removeLiquidityETHWithPermit(
  address token,
  uint liquidity,
  uint amountTokenMin,
  uint amountETHMin,
  address to,
  uint deadline,
  bool approveMax, uint8 v, bytes32 r, bytes32 s
) external returns (uint amountToken, uint amountETH);
```

Function description:

Remove liquidity from the TRC-20 ⇄ WTRX pool and convert WTRX into TRX. Approval is not needed when removing liquidity due to the existence of permit.

Parameter description:

Parameter	Type	Description
token	address	TRC20 token address
liquidity	uint256	Amount of the liquidity token to be removed
amountTokenMin	uint256	Minimum amount of the token to be received; calculated from the slippage
amountETHMin	uint256	Minimum amount of TRX to be received; calculated from the slippage
to	address	Address receiving token/TRX
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit
approveMax	bool	Whether the amount to be approved by the signature is uint(-1) or liquidity

v	uint8	v component in permit signature
r	bytes32	r component in permit signature
s	bytes32	s component in permit signature

Returns:

amountToken	uint256	Amount of tokens received
amountETH	uint256	Amount of TRX received

7. removeLiquidityETHSupportingFeeOnTransferTokens

```
function removeLiquidityETHSupportingFeeOnTransferTokens(
  address token,
  uint liquidity,
  uint amountTokenMin,
  uint amountETHMin,
  address to,
  uint deadline
) external returns (uint amountETH);
```

Function description:

Remove liquidity from the TRC-20 ⇄ WTRX pool and convert WTRX into TRX. This function is applicable to tokens that charge transfer fees (or deflationary tokens). Approval is needed before the removal of liquidity.

Parameter description:

Parameter	**Type**	**Description**
token	address	TRC20 token address
liquidity	uint256	Amount of liquidity tokens to be removed
amountTokenMin	uint256	Minimum amount of the token needed; calculated from the slippage

amountETHMin	uint256	Minimum amount of TRX needed; calculated from the slippage
to	address	Address receiving token/TRX
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

Returns:

amountETH	uint256	Amount of TRX received

8. removeLiquidityETHWithPermitSupportingFeeOnTransferTokens

```
function removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(
  address token,
  uint liquidity,
  uint amountTokenMin,
  uint amountETHMin,
  address to,
  uint deadline,
  bool approveMax, uint8 v, bytes32 r, bytes32 s
) external returns (uint amountETH);
```

Function description:

Remove liquidity from the TRC-20 ⇄ WTRX pool and convert WTRX into TRX. This function is applicable to tokens that charge transfer fees. Approval is not needed when removing liquidity due to the existence of permit.

Parameter description:

Parameter	**Type**	**Description**
token	address	TRC20 token address
liquidity	uint256	Amount of liquidity tokens to be removed
amountTokenMin	uint256	Minimum amount of the token needed; calculated from the slippage

amountETHMin	uint256	Minimum amount of TRX needed; calculated from the slippage
to	address	Address receiving token/TRX
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit
approveMax	bool	Whether the amount to be approved by the signature is uint(-1) or liquidity
v	uint8	v component in permit signature
r	bytes32	r component in permit signature
s	bytes32	s component in permit signature

Returns:

amountETH	uint256	Amount of TRX received

9. swapExactTokensForTokens

```
function swapExactTokensForTokens(
  uint amountIn,
  uint amountOutMin,
  address[] calldata path,
  address to,
  uint deadline
) external returns (uint[] memory amounts);
```

Function description:

This parameter is used to swap a specified amount of the input token into the maximum amount of the output token possible. The first element of the parameter path is the address of the input token and the last element is that of the output token. Any elements in between represent the trading pairs. Approval is needed before the swap.

Parameter description:

Parameter	Type	Description
amountIn	uint256	Amount of the input token
amountOutMin	uint256	Minimum amount of the output token needed; calculated from the slippage
path	address[] calldata	A group of token addresses with path.length equal to or larger than 2; all trading pools consisting of adjacent tokens must exist and have liquidity
to	address	Address receiving the output token
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

Returns:

amounts	uint256[] memory	Amount of the input token and all consequent output token

10. swapExactETHForTokens

```
function swapExactETHForTokens(
  uint amountOutMin,
  address[] calldata path,
  address to,
  uint deadline
) external payable returns (uint[] memory amounts);
```

Function description:

This parameter is used to swap a specified amount of TRX into the maximum amount of the output token possible. The first element of the parameter path is the WTRX address and the last element is that of the output token. Any elements in between represent the trading pairs.

Parameter description:

Parameter	Type	Description
msg.value	uint256	Amount of TRX sent
amountOutMin	uint256	Minimum amount of the output token needed; calculated from the slippage
path	address[] calldata	A group of token addresses with path.length equal to or larger than 2; all trading pools consisting of adjacent tokens must exist and have liquidity
to	address	Address receiving the output token
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

Returns:

amounts	uint256[] memory	Amount of the input token and all consequent output token

11. swapTokensForExactETH

```
function swapTokensForExactETH(
  uint amountOut,
  uint amountInMax,
  address[] calldata path,
  address to,
  uint deadline
) external returns (uint[] memory amounts);
```

Function description:

This parameter is used to swap the minimum amount of input token possible into a specific amount of TRX. The first element of the parameter path is the address of the input token, and the last element the WTRX address. Any elements in between represent the pairs to be traded. Approval is needed before the swap. If the to address is a smart contract, it has to be able to receive TRX.

Parameter description:

Parameter	**Type**	**Description**
amountOut	uint256	Amount of TRX to be received
amountInMax	uint256	Maximum amount of the input token needed; calculated from the slippage
path	address[] calldata	A group of token addresses with path.length equal to or larger than 2; all trading pools consisting of adjacent tokens must exist and have liquidity
to	address	Address receiving TRX
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

Returns:

amounts	uint256[] memory	Amount of the input token and all consequent output token

12. swapExactTokensForETH

```
function swapExactTokensForETH(
  uint amountIn,
  uint amountOutMin,
  address[] calldata path,
  address to,
  uint deadline
) external returns (uint[] memory amounts);
```

Function description:

This parameter is used to swap a specified amount of input token into the maximum amount of TRX possible. The first element of the parameter path is the address of the input token, and the last element the WTRX address. Any elements in between represent the pairs to be traded. If the to address is a smart contract, it has to be able to receive TRX.

Parameter description:

Parameter	Type	Description
amountIn	uint256	Amount of the input token
amountOutMin	uint256	Minimum amount of the output token needed; calculated from the slippage
path	address[] calldata	A group of token addresses with path.length equal to or larger than 2; all trading pools consisting of adjacent tokens must exist and have liquidity
to	address	Address receiving TRX
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

Returns:

amounts	uint256[] memory	Amount of the input token and all consequent output token

13. swapETHForExactTokens

```
function swapETHForExactTokens(
  uint amountOut,
  address[] calldata path,
  address to,
  uint deadline
) external payable returns (uint[] memory amounts);
```

Function description:

This parameter is used to swap the minimum amount of input token possible into a specific amount of TRX. The first element of the parameter path is the WTRX address and the last element is that of the output token. Any elements in between represent the trading pairs. If the to address is a smart contract, it has to be able to receive TRX.

Parameter description:

Parameter	Type	Description
amountOut	uint256	Amount of output token to be received
msg.value (amountInMax)	uint256	Maximum amount of TRX to be paid; calculated from the slippage
path	address[] calldata	A group of token addresses with path.length equal to or larger than 2; all trading pools consisting of adjacent tokens must exist and have liquidity
to	address	Address receiving the output token
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

Returns:

amounts	uint256[] memory	Amount of the input token and all consequent output token

14. swapExactTokensForTokensSupportingFeeOnTransferTokens

```
function swapExactTokensForTokensSupportingFeeOnTransferTokens(
  uint amountIn,
  uint amountOutMin,
  address[] calldata path,
  address to,
  uint deadline
) external;
```

Parameter description:

This parameter is used to swap a specified amount of the input token into the maximum amount of the output token possible. The first element of the parameter path is the address of the input token, and the last element that of the output token. Any elements in between represent the pairs to be traded. This function is applicable to tokens that charge transfer fees. Approval is needed before the swap.

Parameter description:

Parameter	Type	Description
amountIn	uint256	Amount of the input token
amountOutMin	uint256	Minimum amount of the output token needed; calculated from the slippage
path	address[] calldata	A group of token addresses with path.length equal to or larger than 2; all trading pools consisting of adjacent tokens must exist and have liquidity
to	address	Address receiving the output token
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

15. swapExactETHForTokensSupportingFeeOnTransferTokens

```
function swapExactETHForTokensSupportingFeeOnTransferTokens(
  uint amountOutMin,
  address[] calldata path,
  address to,
  uint deadline
) external payable;
```

Parameter description:

This parameter is used to swap a specified amount of TRX into the maximum amount of the output token possible. The first element of the parameter path is the WTRX address, and the last element that of the output token. Any elements in between represent the pairs to be traded.This function is applicable to tokens that charge transfer fees (or deflationary tokens).

Parameter description:

Parameter	**Type**	**Description**
msg.value (amountIn)	uint256	Amount of TRX sent
amountOutMin	uint256	Minimum amount of the output token needed; calculated from the slippage
path	address[] calldata	A group of token addresses with path.length equal to or larger than 2; all trading pools consisting of adjacent tokens must exist and have liquidity
to	address	Address receiving the output token
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit

16. swapExactTokensForETHSupportingFeeOnTransferTokens

```
function swapExactTokensForETHSupportingFeeOnTransferTokens(
  uint amountIn,
  uint amountOutMin,
  address[] calldata path,
  address to,
  uint deadline
) external;
```

Parameter description:

This parameter is used to swap a specified amount of input token into the maximum amount of TRX possible. The first element of the parameter path is the address of the input token, and the last element the WTRX address. Any elements in between represent the pairs to be traded. This function is applicable to tokens that charge transfer fees (or deflationary tokens). If the to address is a smart contract, it has to be able to receive TRX.

Parameter description:

Parameter	**Type**	**Description**
amountIn	uint256	Amount of the input token
amountOutMin	uint256	Minimum amount of the output token needed; calculated from the slippage
path	address[] calldata	A group of token addresses with path.length equal to or larger than 2; all trading pools consisting of adjacent tokens must exist and have liquidity
to	address	Address receiving TRX
deadline	uint256	Unix timestamp; transaction will revert if exceeding this time limit