**Smart Contract Documentation: Conditional Order Protocol**

**Overview**

The Conditional Order Protocol is a decentralized Ethereum smart contract that facilitates conditional orders for buying and selling assets. It provides a robust framework for creating, executing, and managing orders based on various conditions. This documentation offers a detailed guide to understanding the contract and includes test cases to ensure its functionality.

**Table of Contents**

Introduction

Contract Structure

Contract Methods

placeOrder

executeOrder

cancelOrder

transferOrderOwnership

Conditions

Test Cases

License

Acknowledgments

**Introduction**

The Conditional Order Protocol smart contract is designed to provide a decentralized platform for creating and managing conditional orders. It enables users to specify conditions for executing orders, such as asset prices, event triggers, and time-based constraints.

**Contract Structure**

**Enums**

OrderStatus: Defines the status of an order, which can be one of the following:

Open: The order is open and can be executed.

Executed: The order has been successfully executed.

Canceled: The order has been canceled by the owner.

**Structs**

Order: Represents an order with the following attributes:

user: The address of the order owner.

assetToBuy: The address of the asset to buy.

amountToBuy: The amount of the asset to buy.

assetToSell: The address of the asset to sell.

amountToSell: The amount of the asset to sell.

conditionIds: An array of condition IDs that must be met for execution.

status: The current status of the order.

Condition: Represents a condition with two attributes:

conditionType: A bytes32 value representing the type of the condition.

conditionData: A bytes32 value containing condition-specific data.  
  
**Contract Methods**

**placeOrder**

-function placeOrder(

address \_assetToBuy,

uint256 \_amountToBuy,

address \_assetToSell,

uint256 \_amountToSell,

uint256[] memory \_conditionIds

) public

**Allows a user to place an order with specified parameters and conditions.**

executeOrder

function executeOrder(uint256 \_orderId) public

Executes an open order if all specified conditions are met.

cancelOrder

function cancelOrder(uint256 \_orderId) public onlyOrderOwner(\_orderId)

Allows the owner of an open order to cancel it.

transferOrderOwnership

solidity

Copy code

function transferOrderOwnership(uint256 \_orderId, address \_newOwner) public onlyOrderOwner(\_orderId)

Permits the owner of an order to transfer its ownership to another address.

Conditions

The smart contract allows users to define custom conditions, which can be referenced in the placeOrder method. Conditions are created using the addCondition function, which stores condition type and data.

**Test Cases**

Prerequisites

Node.js

Hardhat

Chai

Installation

Clone this repository.

Navigate to the project directory.

Install the project dependencies:

bash

Copy code

npm install

Running Tests

Use the following command to run the test cases:

bash

npx hardhat test