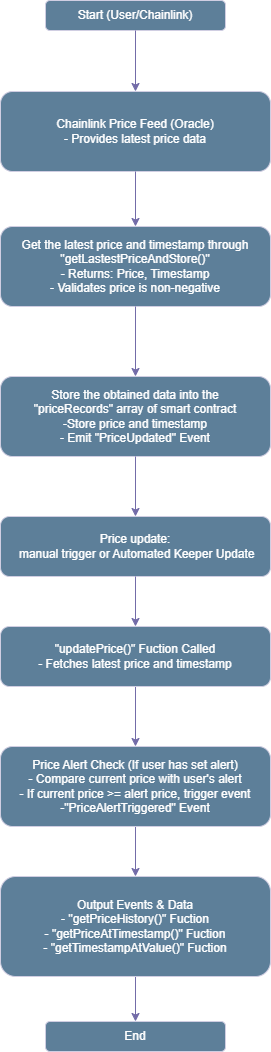
**Back-end code analysis**

1. System architecture and design

1.1Smart Contract Structure



The contract is based on CryptoPriceTracker as the core contract, inheriting the Ownable (ownership of the managed contract) and KeeperCompatibleInterface (support for Chainlink Keeper automation) contracts.

Use Chainlink's AggregatorV3Interface interface to obtain cryptocurrency prices from a decentralized oracle.

Introduce a scheduled update mechanism to ensure the timeliness of price information.

1.2 Key functional modules:

Price record storage: Use the PriceData structure to store the price and timestamp of each price update.

1.3 User price alert

Users can set price alerts, and when the price reaches or exceeds the preset value, the contract will trigger an alert event.

1.4 Historical price query

Support users to query historical price records.

Price inquiry and update mechanism: Through Chainlink's Keeper service, contracts automatically update prices on a regular basis.

1. Implementation of core functions

2.1 Price acquisition and storage

Use Chainlink's latestRoundData() method to obtain the latest cryptocurrency prices. Store prices and their timestamps for easy retrieval and analysis in the future. Trigger events so that external systems can monitor price updates in real-time.

2.2 Price update mechanism:

The price is updated every 10 minutes (configurable) to ensure data timeliness. Use Chainlink Keeper to trigger the updatePrice() function in the contract at regular intervals to achieve automatic updates.

2.3 Price alert function

Users can set their own price alerts, which will be triggered by the contract when the market price exceeds the set threshold.

2.4 Historical price inquiry:

Users can query past price records, supporting queries by timestamp or price.

1. Security and Permission Management

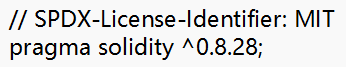
3.1 Contract ownership control

Inherited from Ownable contracts, ensuring that only the contract owner can perform certain sensitive operations (such as modifying contract settings).

3.2Contract function permissions

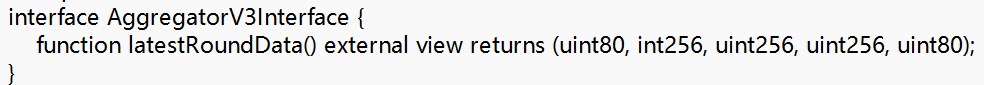
Most functions are open to all users, while some key operations (such as contract management, price update settings) are only available to contract owners.

1. Code Explanation
   1. SPDX License and Pragma Directive



The contract uses the MIT license to indicate the open source code agreement of the contract. The version of the Solidity compiler used for this contract is ^0.8.28 .

* 1. Chainlink Price Feed Interface



This part is used to interact with the price of price feeding with Chainlink. Get the latest round of price data through LatestroundDdata function, including: Round ID(uint80), Price (int256), Timestamp(uint256), other related data(unint256), Round ID(uint80).

* 1. Import the Chainlink Keeper interface



This line of code imports the Chainlink Keeper interface to interact with the automated service of ChainLink.

* 1. Ownable Contract

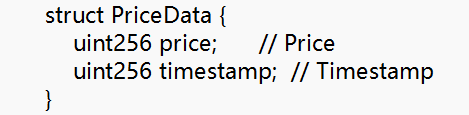


Having a contract is the basic contract for management contract ownership. It provides functions related to the contract owner and uses the contract template provided by OpenZeppelin.

The "CryptoPriceTracker" contract we wrote inherited the Ownable contract. This is convenient for the management of the CryptoPriceTracker contract and improves the security of the contract.

* 1. Cryptopricetracker contract

4.5.1 PriceData



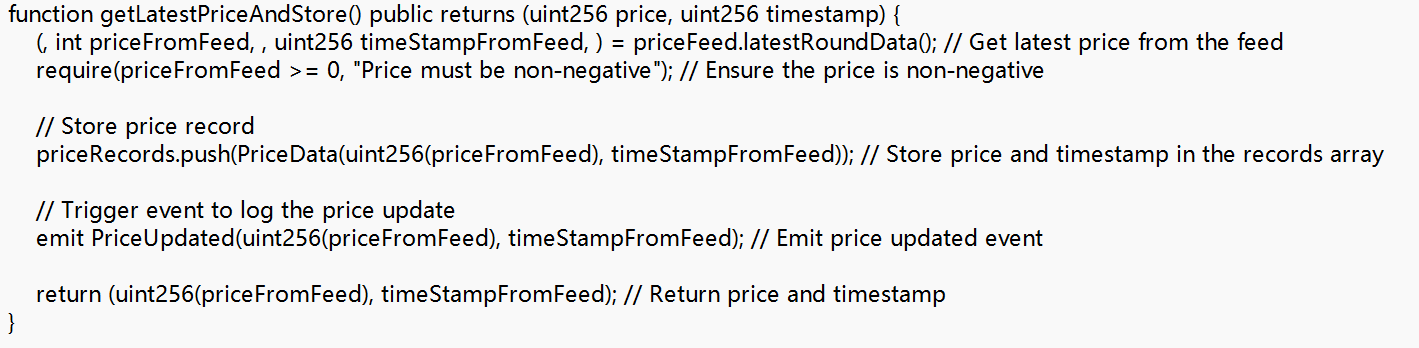
The PriceData structure is used to store prices and timestamps.

4.5.2 PriceRecords



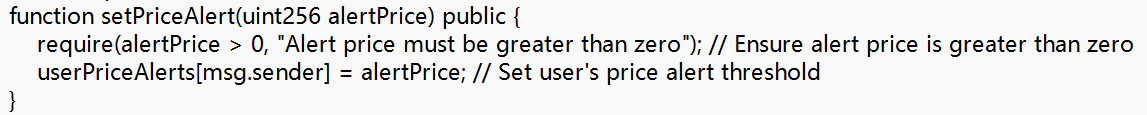
PriceRecords array is used to store price data of all historical, and each element is an object of "PriceData" type.

4.5.3getLastestPriceAndStore



This function calls ChainLink's LatestRoundData () function to get the latest price and timestamp. It first checks whether the price is positive, and then stores the correct price and timestamp into the PriceRecords array. Finally it triggers the PriceUpDated event, record the price update information, and return the latest price and timestamp.

4.5.4 setPriceAlert



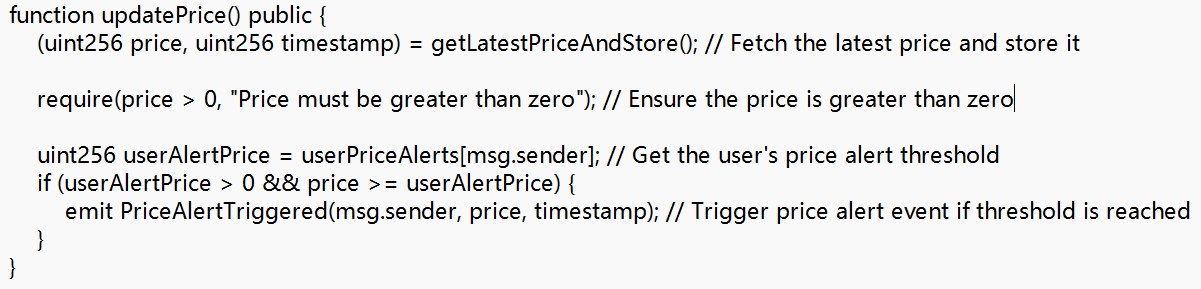
Users can set up a warning threshold by passing a AlertPrice variables greater than zero to the Setpricealert function. The setting threshold after the setting will be associated to the user's address.

4.5.5PriceAlertTriggered



This incident will conduct alarm when the price exceeds the alert value of the user. The user's address, current price and timestamp are recorded in the incident.

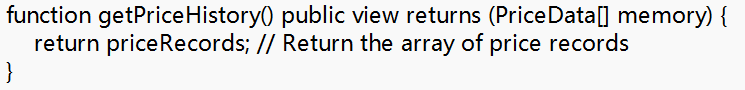
4.5.6 updatePrice



The function of this function is to update the price. First of all, it will call the getLatestPriceAndStore function to get the latest price data and store it.

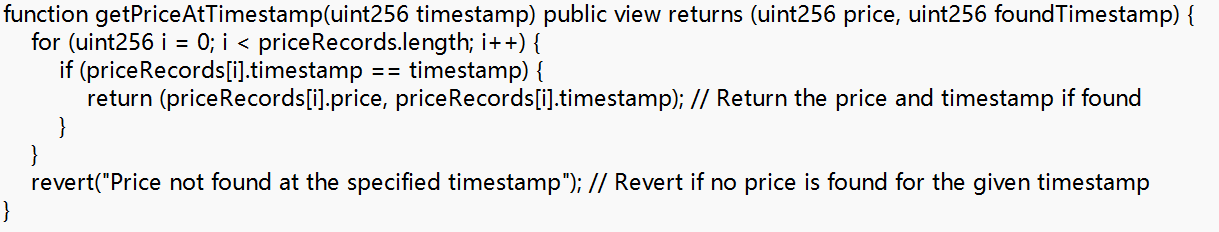
Then it will check whether the user sets the price alert. If the alarm is set and the current price is greater than or equal to the alarm threshold, the PriceAlertTriggered event is triggered.

4.5.7 getPriceHistory



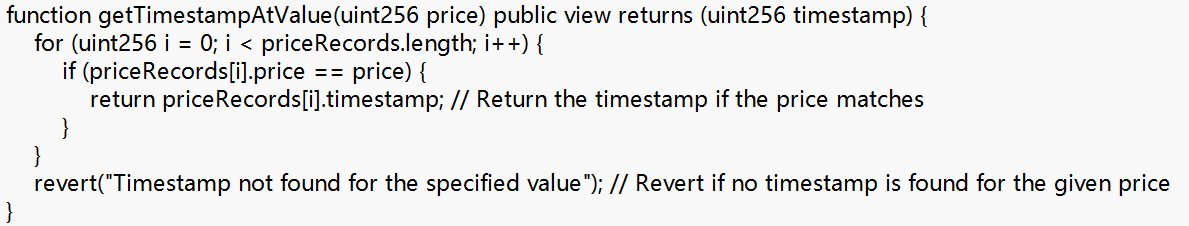
The function of this function is to return all historical price records for users to check.

4.5.8 getPriceAtTimestamp



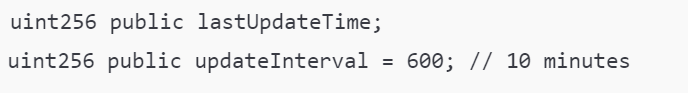
The user can pass a timestamp to the function, and the function will find the corresponding price data in the PriceRecords array, and then returns these data. If the data is not found, then it will throw an exception.

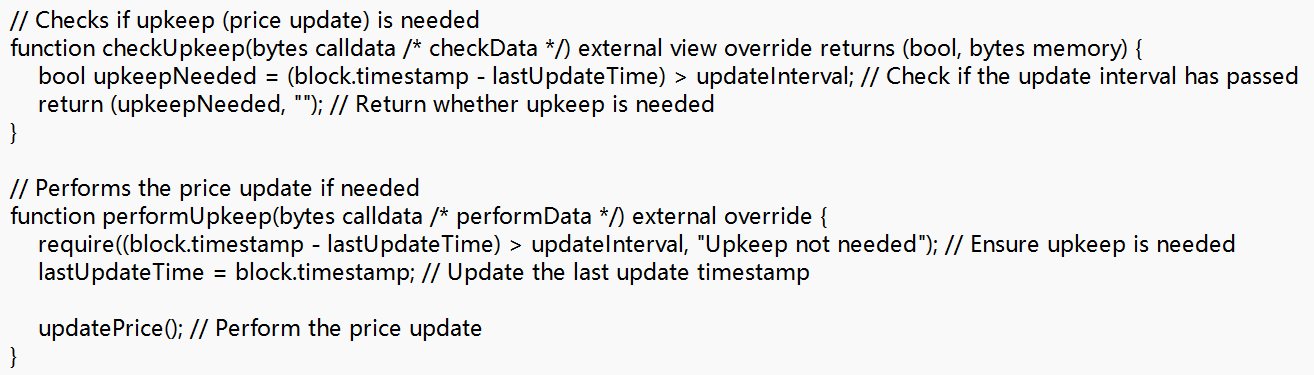
4.5.9 getTimestampAtValue



The user can pass a price value to the function, and the function will find the corresponding timestamp in the PriceRecords array, and then returns these data. If the data is not found, then it will throw an exception.

* 1. ChainLink Keeper related functions





The CryptoPricetracker contract also integrates the "Chainlink Keeper" service in order to automatically update the price regularly.

1. System optimization and expansion

Automation and Timer Function: Use Chainlink Keeper for automated scheduled task scheduling, ensuring that the system can run sustainably without human intervention.

Storage optimization: Storing price records on the chain may increase on chain storage costs. Optimizing storage methods, regularly cleaning outdated data, or introducing external storage solutions such as IPFS can effectively reduce costs.

Multiple asset support: Currently, the contract supports price tracking of a single asset, and in the future, it can be expanded to support price monitoring of multiple encrypted assets (such as BTC, ETH, etc.).