SubscriptionLicense Contract Documentation

This document describes the SubscriptionLicense contract, which enables the creation and management of subscription-based licenses for products or services.

Overview

The SubscriptionLicense contract allows users to create and manage recurring subscriptions using ERC20 tokens. It provides functionality for executing, checking the status, and canceling subscriptions. The contract also supports a grace period and custom gas price settings.

Contract Details

Libraries

- ECDSA.sol: Used for Ethereum message signing and signature recovery.
- ERC20.sol: Standard ERC20 token contract for managing tokens.
- SafeMath.sol: Provides arithmetic operations without overflow.

Events

- ExecuteSubscription: Emitted when a subscription is executed successfully.
- CancelSubscription: Emitted when a subscription is canceled successfully.

Constructor

The constructor sets the following initial values:

- requiredToAddress: The address to which the subscription payments will be sent.
- requiredTokenAddress: The address of the ERC20 token used for the subscription.
- requiredTokenAmount: The amount of tokens to be transferred for each subscription period.
- requiredPeriodSeconds: The duration of each subscription period in seconds.
- requiredGasPrice: The gas price for executing the subscription.
- i_licenseName: A name for the subscription license.

Functions

- **isSubscriptionActive(subscriptionHash, gracePeriodSeconds)**: Checks if the subscription is active, taking into account the grace period.
- **getSubscriptionHash(from, to, tokenAddress, tokenAmount, periodSeconds, gasPrice, nonce)**: Generates a unique hash for the subscription based on the provided parameters.
- **getSubscriptionSigner(subscriptionHash, signature)**: Returns the address of the signer of a subscription.

- **isSubscriptionReady(from, to, tokenAddress, tokenAmount, periodSeconds, gasPrice, nonce, signature)**: Checks if the subscription is ready to be executed.
- **cancelSubscription(from, to, tokenAddress, tokenAmount, periodSeconds, gasPrice, nonce, signature)**: Cancels a subscription.
- **executeSubscription(from, to, tokenAddress, tokenAmount, periodSeconds, gasPrice, nonce, signature)**: Executes a subscription by transferring the required tokens and updating the next valid timestamp.
- **checkSuccess()**: Checks if a token transfer is successful.

Getter Functions

- **getRequiredToAddress()**: Returns the required to address.
- **getRequiredTokenAddress()**: Returns the required token address.
- **getRequiredTokenAmount()**: Returns the required token amount.
- **getRequiredPeriodSeconds()**: Returns the required period in seconds.
- **getRequiredGasPrice()**: Returns the required gas price.
- **getILicenseName()**: Returns the subscription license name.

How to Use

1. Deploy the SubscriptionLicense contract with the required parameters.
2. For a user to subscribe, they must first approve the contract to spend the required token amount.
3. The user can then execute the subscription by calling the executeSubscription function with the correct parameters and signature.
4. The user can check the status of their subscription by calling the isSubscriptionActive function.
5. The user can cancel their subscription by calling the cancelSubscription function.
6. Please note that the user must have sufficient token balance and allowance for the subscription to be executed successfully."