# Function Call Details for Interacting with the Legacy Smart Contract

This documentation provides instructions for frontend developers on how to interact with the Legacy smart contract to activate subscriptions, change heir addresses, renew subscriptions, and handle deposits and withdrawals. Below are the details for each function call.

## 1. Activate Subscription

There are two cases for activating a subscription, depending on whether Ethereum or stable coins are used.

### a. Activated by Ethereum

Code:

const etherAmount = (await legacy.calculateETHFee()) + ethers.parseEther(0.1);  
const tx = await legacy.activateSubscription(heir.address, ethers.ZeroAddress, {  
 value: etherAmount, });

### b. Activated by Stable Coin

Code:

const feeInStableCoin = await legacy.feeInStableCoin();  
await stableCoin.approve(await legacy.getAddress(), feeInStableCoin);  
const tx = await legacy.activateSubscription(heir.address, await usdt.getAddress());

## 2. Change Heir Address

To change the heir address for an active subscription, use the following function:

Code:

const tx = await legacy.changeHeir(subscriptionId, newHeirAddress);

## 3. Renew Subscription

There are two methods to renew subscriptions, depending on whether Ethereum or stable coins are used.

### a. Renew with Ethereum

Code:

const ethFee = await legacy.calculateETHFee();  
// Renew the subscription with ETH  
const tx = await legacy.renewSubscription(ethers.ZeroAddress, subscriptionId, {  
 value: ethFee,  
});

### b. Renew with Stable Coin

Code:

const baseFeeUSD = await legacy.baseFeeUSD();  
await usdt.approve(await legacy.getAddress(), baseFeeUSD);  
// Renew the subscription with USDT  
const tx = await legacy.renewSubscription(await usdt.getAddress(), subscriptionId);

## 4. Deposit Funds

Users can deposit Ether, ERC20 tokens, or both simultaneously. Below are examples for each case:

### a. Deposit Ether Only

Code:

const tx = await legacy.depositFunds([], [], subscriptionId, { value: ethAmount });

### b. Deposit ERC20 Tokens Only

Code:

***Note: Approve each token with respective amount in a batch or separately before calling the following function.***

const tx = await legacy.depositFunds(arrayOfTokens, arrayOfAmounts, subscriptionId);

### c. Deposit Both Ether and ERC20 Tokens

Code:

***Note: Approve each token with respective amount in a batch or separately before calling the following function.***

const tx = await legacy.depositFunds(arrayOfTokens, arrayOfAmounts, subscriptionId, {  
 value: ethAmount,  
});

## 5. Withdraw Funds

There are three cases for withdrawing funds: Ether, ERC20 tokens, or both.

### a. Withdraw Ether Only

Code:

await legacy.withdrawFunds(ethersAmount, [], [], subscriptionId);

### b. Withdraw ERC20 Tokens Only

Code:

await legacy.withdrawFunds(0, arrayOfTokens, arrayOfAmounts, subscriptionId);

### c. Withdraw Both Ether and ERC20 Tokens

Code:

await legacy.withdrawFunds(ethersAmount, arrayOfTokens, arrayOfAmounts, subscriptionId);

## 6. Retrieve Active Subscriptions of a User

To retrieve the active subscription IDs associated with a user, use the following function:

Code:

await legacy.getActiveIdsOfUser(userAddress);

## 7. Retrieve Subscription Details by ID

To get the detailed information of a specific subscription by its ID, use:

Code:

await legacy.getSubscriptionById\_withTokens(userAddress, id);

# Conclusion

This documentation outlines the key function calls for interacting with the Legacy smart contract from the frontend. Make sure to handle transactions and approvals securely, and adapt the functions as per your application's logic.