

# PullPayment Registry Contract

## Introduction

The purpose of the PullPayment Registry contract is to have the central directory for all the pullPayment contracts. It allows the admin to register the pullPayment and access the address of pullPayments in one place.

It also stores the upkeep information, batch size for the subscriptions and maintains the list of low balance subscriptions of pull payments.

## Contract version

pragma solidity 0.8.0

## Contract constructor

In this contract the initialize() method is used in place of the constructor to allow the contract to be upgradable via proxy.

### Method Name:

- initialize

### Method Usage:

- The initialize method is used to initialize the PullPayment Registry contract and transfers the ownership of the registry to the admin who deploys the registry.
- Sets the default BATCH\_SIZE for the subscriptions list.

### Method Signature:

- function initialize() external;

### Method Parameters:

- No parameters required.

# Contract Methods

## grantExecutor()

Method name:

- `grantExecutor`

Method detail:

- This method allows the registry owner to grant any wallet address to be the executor for executing `pullPayments` using the `execute` method of the `Executor` contract.
- Generally we grant the `pullPayment` contracts to execute the `pullPayments` and add the low balance subscriptions to the list.
- Only the owner can grant the executor role to any address.

Method signature:

- ```
function grantExecutor(address _executor)
    external
    override;
```

Method parameters:

**1. `_executor`:**

Indicates the wallet address to whom we allow to be an executor.

Returned data:

- This method does not return any data.

# revokeExecutor()

Method name:

- revokeExecutor

Method detail:

- This method allows the owner of pullPayment registry to revoke the executor permission from the wallet address.
- Revoking the executor role will not allow the user to execute the pullPayment on the Executor contract and to add/remove the low balance subscriptions.

Method signature:

- function revokeExecutor(address \_executor)  
external  
override

Method parameters:

- **\_executor:**
  - Indicates the executor address whose we remove the executor role.

Returned data:

- This method does not return any data.

## addPullPaymentContract()

Method name:

- addPullPaymentContract

Method detail:

- This method allows an owner to register the pullPayment on registry and grant the pullPayment contract and executor role.
- Only the owner can register the pullPayments.
- This method associates the given pullPayment contract address with the given identifier.

Method signature:

- function addPullPaymentContract(string calldata \_identifier, address \_addr)  
external  
override

Method parameters:

- **\_identifier:**
  - Indicates the name of the pullPayment contract that we want to register.
- **\_addr:**
  - Indicates the address of the pullPayment contract.

Returned data:

- This method does not return any data.

## addLowBalanceSubscription()

Method name:

- addLowBalanceSubscription

Method detail:

- This method allows the granted executor contracts to add the low balance subscriptions in the list maintained for that executor.

Method signature:

- function addLowBalanceSubscription(uint256 \_subscriptionId)  
public  
virtual

Method parameters:

- **\_subscriptionId:**
  - Indicates the low balance subscription id.

Returned data:

- This method does not return any data.

## removeLowBalanceSubscription()

Method name:

- removeLowBalanceSubscription

Method detail:

- This method allows the granted executor contracts to remove the low balance subscriptions from the list maintained for that executor.

Method signature:

- function removeLowBalanceSubscription(uint256 \_subscriptionId)  
public  
virtual

Method parameters:

- **\_subscriptionId:**
  - Indicates the low balance subscription id.

Returned data:

- This method does not return any data.

## setUpkeepId()

Method name:

- setUpkeepId

Method detail:

- This method allows the admin to add the upkeep id for the pull payment contract.

Method signature:

- function setUpkeepId(address upkeepAddress, uint256 upkeepId) external virtual

Method parameters:

- **upkeepAddress:**
  - Indicates the pullpayment contracts address.
- **upkeepId:**
  - Indicates the upkeepId for the registered upkeep.
  - We can get this upkeep id after registering the pull payment contract as upkeep on the chainlink keepers dashboard.

Returned data:

- This method does not return any data.

## updateBatchSize()

Method name:

- updateBatchSize

Method detail:

- This method allows the admin to update the batch size for the subscription list.
- This BATCH\_SIZE is used by the upkeeps in order to get the maximum number of subscription ids for the execution.
- Ex. If the BATCH\_SIZE is 20. Then the **checkUpkeep()** method can return a maximum 20 subscription ids for the execution.

Method signature:

- function updateBatchSize(uint256 \_batchSize) external

Method parameters:

- **\_batchSize:**
  - Indicates the new batch size

Returned data:

- This method does not return any data.

# getPPAddressForOrDie()

Method name:

- getPPAddressForOrDie

Method detail:

- This method retrieves the contract's address associated with the given identifier hash.
- If it does not find the contract's address for a given identifier hash, it throws an error.

Method signature:

- function getPPAddressForOrDie(bytes32 \_identifierHash)  
    external  
    view  
    override  
    returns (address)

Method parameters:

- **\_identifierHash:**
  - Indicates the contract name in bytes32 format.

Returned data:

- This method returns the address of the pullPayment contract associated with the identifier.



## getPPAddressFor()

Method name:

- getAddressFor

Method detail:

- This method retrieves the pullPayment contract address for the given identifier hash.
- If it does not get the address for the given identifier, it returns the default zero address.

Method signature:

- function getPPAddressFor(bytes32 \_identifierHash)  
    external  
    view  
    override  
    returns (address)

Method parameters:

- **\_identifierHash:**
  - Indicates the pullPayment contract name in bytes32 format.

Returned data:

- This method returns the pullPayment contract address if it exists or it returns the zero-address.

## getPPAddressForStringOrDie()

Method name:

- `getPPAddressForStringOrDie`

Method detail:

- This method returns the pullPayment contract address for the given contract name.
- It throws an error if it does not find the address for the given contract name.

Method signature:

- `function getPPAddressForStringOrDie(string calldata _identifier)`  
external  
view  
override  
returns (address)

Method parameters:

- **`_identifier`:**
  - Indicates the pullPayment contract name in string format.

Returned data:

- This method returns the pullPayment contract address if it exists.

## getPPAddressForString()

Method name:

- getPPAddressForString

Method detail:

- This method retrieves the address of the pullPayment contract for a given contract name.
- It returns the zero-address if it does not find the contract address associated with the given contract name.

Method signature:

- function getPPAddressForString(string calldata \_identifier)  
    external  
    view  
    override  
    returns (address)

Method parameters:

- **\_identifier:**
  - Indicates the name of the pullPayment contract in string format.

Returned data:

- This method returns the address of the pullPayment contract.

## isExecutorGranted()

Method name:

- isExecutorGranted

Method detail:

- This method tells you whether the given address is an executor or not.

Method Signature:

- function isExecutorGranted(address \_executor)  
external  
view  
override  
returns (bool);

Method parameters:

- **\_executor:**
  - Indicates the address of a pullPayment contract.

Returned data:

- This method returns true if the given address is an executor otherwise it returns false.

# Events

## RegistryUpdated

Event Name:

- RegistryUpdated

Event detail:

- This event is emitted when a new pullPayment contract is registered using the **addPullPaymentContract()** method.

Event signature:

- event RegistryUpdated(  
    string identifier,  
    bytes32 indexed identifierHash,  
    address indexed addr  
);

Event data:

- **identifier:**
  - Indicates the name of the pullPayment contract.
- **identifierHash:**
  - Indicates the pullPayment contract name in bytes32 format.
- **Addr:**
  - Indicates the address of the pullPayment contract.

## ExecutorGranted

Event name:

- ExecutorGranted

Event detail:

- This event is emitted when the owner grants the executor role to any address using the **grantExecutor()** or **addPullPaymentContract()** method.

Event signature:

- event ExecutorGranted(address executor);

Event data:

- **executor:**
  - Indicates address of the pullPayment contract to whom we granted the executor role.

## ExecutorRevoked

Event name:

- ExecutorRevoked

Event detail:

- This event is emitted when the owner revokes the executor role from any address using the **revokeExecutor()**.

Event signature:

- event ExecutorRevoked(address executor);

Event data:

- **executor:**
  - Indicates address of the pullPayment contract from whom we revoked the executor role.

## Flow of Execution:

1. When we deploy the new pullPayment contract in the ecosystem which executes the pullPayment using the execute() method of the Executor contract, that contract must be registered on the pullPayment Registry.
2. Owner needs to register the new contract on the registry.
3. Owner Grants/revokes the executor role to/ from the pullPayment contract.
4. Get addresses of registered pullPayment contracts.
5. Owner stores the upkeep ids for the pull payment contracts.
6. Upkeeps can add/remove the subscription id to/from the low balance subscription list.