Recurring Dynamic PullPayment

Introduction

The Recurring Dynamic PullPayment simplifies the recurring payments that customers need to do on a regular interval to avail the services. This contract allows customers to subscribe to the billing model and start the recurring payments. It also provides the free trial and paid trial of the billing model.

A Dynamic PullPayment, like a recurring PullPayment. However, in this case, the payment properties (currency, price, name etc.) can be injected straight onto merchant websites and not through the Business Console.

This type of billing model is most suited to merchants who sell tens/hundreds/thousands of products with different descriptions and prices.

Using this contract merchant can create the billing model by providing the payee address and recurring pullPayment type i.e paid trial, free trial or normal recurring pullPayment. The rest of the details are provided dynamically when the customer subscribes to the billing model.

Contract version

pragma solidity 0.8.0

Contract constructor

The Recurring Dynamic Pullpayment contract`s constructor takes one argument i.e the main registry contract address which is used to access the registry methods.

Constructor Signature:

- constructor(address registryAddress) RegistryHelper(registryAddress);

Constructor Parameters:

1. **registryAddress:**
    - indicates the address of the registry contract which keeps the record of all the contract addresses in the ecosystem and the required configurations.

Contract Methods

createBillingModel()

Method name:

- createBillingModel

Method detail:

- This method allows merchants to create a new billing model for their business.
- The unique Id is created for each billing model.
- Merchant can create as many billing models as he wants.
- In this contract, merchants can specify the trial period, the type of the recurring pullPayment, merchant name, unique reference for billing model and the merchant URL.
- The rest of the details are provided dynamically to the customer through the code-snippet.
- The **BillingModelCreated** event is emitted after creating the billing model.

Method signature:

- ```
  function createBillingModel(
    address _payee,
    uint8 _recurringPPType,
    string memory _merchantName,
    string memory _reference,
    string memory _merchantURL
  ) external virtual override returns (uint256 billingModelID)
  ```

Method parameters:

1. **_payee:**
   - Indicates the address of the receiver(merchant) who will receive the pull payments.
2. **_recurringPPType:**
   - Indicates the type of the recurring pullPayment. Here are the ids of recurring pullPayments-
     i. 1- Normal Recurring pullPayment
     ii. 2- Free Trial Recurring PullPayment
     iii. 3- Paid Trial Recurring PullPayment
3. **_merchantName:**
   - Indicates the name of the merchant. It can be kept empty.
4. **_reference:**
   - Indicates the unique reference for the billing model.
   - If it is not passed externally, then the contract creates the unique reference for the billing model.
5. **_merchantURL:**
   - Indicates the merchant`s personal URL. It can be kept empty.

Returned data:

- This method returns the unique billing model Id.
  ```

subscribeToBillingModel()

Method name:

- subscribeToBillingmodel

Method detail:

- This method allows any customer to subscribe to the billing model with a given billing model Id and payment token.
- The remaining details of the billing model are provided dynamically through the code-snippet.
- The remaining details include the billing model name, settlement token, payment amount, frequency of payments, total number of payments, the trial period and the initial amount for the trial period.
- Here, trial period value needs to be specified in case of a free trial and paid trial contract and the initial amount needs to be specified only in case of paid trial pullPayment.
- In this, the type of the recurring pull payment is identified with the help of billing model id.
- The user must approve the payment tokens to the Executor contract before subscribing to the billing model.
- The unique subscription id is created on each subscription.
- This method emits a **NewSubscription** event after subscribing to the billing model.

Method signature:

- ```
  function subscribeToBillingModel(
    uint256 _billingModelID,
    string memory _bmName,
    address _settlementToken,
    address _paymentToken,
    uint256 _paymentAmount,
    uint256 _frequency,
    uint256 _totalPayments,
    uint256 _trialPeriod,
    uint256 _initialAmount,
    string memory _reference
  )
    external
    virtual
    override
    nonReentrant
    returns (uint256 newSubscriptionID)
  ```
  ```

Method parameters:

- **_billingModelID:**
  - Indicates the unique billing model id that the customer wants to subscribe.
- **_bmName:**
  - Indicates the name of the billing model.
- **_settlementToken:**
  - Indicates the settlement token address in which the payee wants to receive the payment.
- **_paymentToken:**
  - Indicates the payment token address in which the customer wants to pay for the subscription after the trial period.
- **_paymentAmount:**
  - Indicates the billing model amount which will be charged during each pullPayment
- **_frequency:**
  - Indicates the time interval in seconds after which the pullPayment is executed.
- **_totalPayments:**
  - Indicates the total number of pullPayments that customer needs to complete after the fixed interval of time.
- **_trialPeriod:**
  - Indicates the trial period in seconds for the subscription. Trial period is specified only for free trial and paid trial of recurring pullPayments.
- **_initialAMount:**
  - Indicates the initial amount that will be charged for the paid trial type of recurring pullPayment.
- **_reference:**
  - Indicates the unique reference for the subscription. If external reference is not given the contract automatically generates new reference for the subscription.

Returned data:

- This method returns the unique subscription id after subscription.

executePullPayment()

Method name:

- executePullPayment

Method detail:

- This method allows anyone to execute the pullPayment for the given subscription id.
- It ensures pullPayment execution time is correct i.e trial period has ended, subscription is not cancelled and no. of payments are not exceeded.
- Internally this method calls the execute method of the Executor contract to transfer the tokens from customer to merchant.
- It updates the next payment timestamp, number of payments remaining, pullPayment execution timestamp and last payment timestamp.
- The unique pullPayment id is created for each pullPayment and added to the list of pullpayment ids of subscription.
- This method emits the **PullPaymentExecuted** event after executing pullpayment.

Method signature:

- function executePullPayment(uint256 _subscriptionID)
  public override returns (uint256 pullPaymentID)

Method parameters:

- **_subscriptionID:**
  - Indicates the unique subscription id.

Returned data:

- This method returns the unique pullPayment id after pullPayment execution.

cancelSubscription()

Method name:

- cancelSubscription

Method detail:

- This method allows subscribers to cancel their subscription to stop the pull payments.
- Users can cancel subscriptions during the trial period as well.
- It updates the cancel timestamp, adds the address of the account who cancelled the subscription and makes the subscription inactive.
- Only the merchant and subscriber can cancel the subscription.
- Subscription gets automatically cancelled after a specific period of time when there are not enough tokens in the subscriber's account for the pullpayment.
- This method emits a **SubscriptionCancelled** event after cancelling the subscription.

Method signature:

- function cancelSubscription(uint256 _subscriptionID)
  public
  virtual
  override
  returns (uint256 subscriptionID)

Method parameters:

- **_subscriptionID:**
  - Indicates the unique subscription id.

Returned data:

- This method returns the id of the cancelled subscription.

editBillingModel()

Method name:

- editBillingModel

Method detail:

- This method allows the billing model creator i.e payee to edit the details of the billing model.
- It allows payee to update the billing model name, payee address, merchant name and merchant url.
- Only the owner of the billing model can edit the billing model details.

Method signature:

- function editBillingModel(
  uint256 _billingModelID,
  string memory _newName,
  string memory _newMerchantName,
  string memory _newMerchantURL
)
  public
  virtual
  override
  returns (uint256 billingModelID)

Method parameters:

- **_billingModelID:**
  - Indicates the unique valid billing model id which the merchant wants to edit.
- **_newPayee:**
  - Indicates the new payee address for the billing model.
  - Merchants cannot set Zero address as payee address.
- **_newMerchantName:**
  - Indicates the new merchant name for the billing model. It can be kept empty.
- **_newMerchantURL:**
  - Indicates the new merchant`s personal URL.

Returned data:

- This method returns the id of the billing model whose details are updated.

checkUpkeep()

Method name:

- checkUpkeep

Method detail:

- This method is used to make the Recurring Dynamic Pull Payment contract as keeper compatible contract.
- This method is periodically called by the Upkeep network after 1 block interval of time.
- It returns the encoded data which contains a list of subscription ids and their count which needs to be executed through the performUpkeep() method.
- It also returns the boolean value based on which keeper network decides whether to call performUpkeep() method or not.
- The subscription ids included in the list satisfies the criteria of pullpayment execution.

Method signature:

- function checkUpkeep(bytes calldata checkData)
  external
  view
  override
  returns (bool upkeepNeeded, bytes memory performData)

Method parameters:

- **checkData:**
  - This argument is specified in the upkeep registration so it is always the same for a registered upkeep.

Returned data:

- This method returns the following values
  - **upkeepNeeded:**
    - This boolean value is set to true when there is at least one subscription in the list of subscription ids.
  - **performData:**
    - This is the encoded data in the bytes format which contains the list of subscription ids and the count of subscription ids in the list.

performUpkeep()

Method name:

- performUpkeep

Method detail:

- This method is used to make the Recurring Dynamic Pull Payment contract as keeper compatible contract.
- This method is called by the Keeper network when the checkUpkeep() method returns a boolean true value.
- This method executes the pullpayment for the list of subscriptions provided in performData.
- If the subscriber does not have the enough payment tokens in his account to make the pullpayment, then this method adds that subscription in the list of low balance subscriptions which is stored on the PullPaymentRegistry contract.
- The low balance subscription is not considered for the execution until the extension period ends. The Extension period is the duration in seconds which is given to the subscriber to top up their account.
- Once this extension period is passed, the low balance subscription is added again for the execution.
- If the subscriber still doesn't have the enough amount of payment tokens in his account then it cancels the subscription.
- If the subscriber has top up his account during the extension period, then it removes the subscription from the low balance subscription ids and continues the pullpayment.

Method signature:

- function performUpkeep(bytes calldata performData) external override

Method parameters:

- **performData:**
  - This is the encoded value which contains the list of subscription ids and their count which needs to be executed.
  - The value of this performData is returned by the checkUpkeep() method.

Returned data:

- This method does not return anything.

getSubscriptionIds()

Method name:

- getSubscriptionIds

Method detail:

- This method is used to get the list of subscription ids and their count whose pull payment should be executed.
- It checks every subscription for the pullpayment criteria. it adds the subscription in the list if it satisfies the condition of pullpayment.
- The criteria for adding the subscription in the list is -
    - Subscription is not added in the list of low balance subscriptions.
    - Subscription is added in the low balance subscription list but has passed the extension period.
    - Adheres to pull payment execution criteria. i.e
        - Remaining number of payments are not zero
        - Subscription is not cancelled
        - Next payment time is in the past.
        - Subscription is not in trial period
- This returns the subscription ids in batches to avoid the out of gas errors while execution.

Method signature:

- function getSubscriptionIds()
  public
  view
  returns (uint256[] memory subscriptionIds, uint256 count)

Method parameters:

- N/A

Returned data:

- This method returns the-
    - List of subscription ids
    - Count of subscriptions in the list.

isPullPayment()

Method name:

- isPullPayment

Method detail:

- This method tells whether to execute the pullpayment for the given subscription id or not based on the pullpayment execution criteria.
- The pull payment execution criteria is-
    - Remaining number of payments are not zero
    - Subscription is not cancelled
    - Next payment time is in the past.
- Returns true if the subscription passes the execution criteria.

Method signature:

- function isPullpayment(uint256 _subscriptionId) public view returns (bool)

Method parameters:

- **_subscriptionId:**
    - Indicates the subscription id.

Returned data:

- This method returns true if the subscription passes the execution criteria otherwise returns false.

getBillingModel()

Method name:

- getBillingModel

Method detail:

- This method retrieves all the details of the specified billing model that is stored on blockchain while creating the billing model. i.e payee address, recurring pullPayment type, merchant name, merchant url, unique reference and billing model creation time.
- Only the payee of a particular billing model can get the subscription ids of the billing model.

Method signature:

- function getBillingModel(uint256 _billingModelID)
  external
  view
  override
  returns (BillingModelData memory bm)

Method parameters:

- **_billingModelID:**
  - Indicates the valid billing model id.

Returned data:

- This method returns the details of the billing model which includes the-
  - Payee address
  - Merchant name
  - The unique reference for the billing model.
  - Merchant URL
  - Type of recurring pullPayment. i.e
    - 1- normal recurring pullPayment
    - 2- free trial recurring pullPayment
    - 3- paid trial recurring pullPayment
  - List of subscription ids
  - Billing model creation time.

getSubscription()

Method name:

- getSubscription

Method detail:

- This method returns the subscription data for a given subscription id.
- Only the payee and subscriber of the billing model can see the pullPayment ids of the subscription.

Method Signature:

- function getSubscription(uint256 _subscriptionID)
  external
  view
  override
  returns (SubscriptionData memory sb)

Method parameters:

- **_subscriptionID:**
  - Indicates the valid subscription id

Returned data:

- This method returns the subscription data which includes the following data-
  - Subscriber address
  - Billing model name
  - Payment amount
  - Settlement token address
  - Payment token address
  - Total number of payments
  - Remaining number of payments
  - Frequency of the payments
  - Start timestamp of subscription
  - Cancelled timestamp of subscription
  - NextPayment timestamp
  - LastPayment timestamp
  - PullPayment IDs(only payee or subscriber gets the ids)
  - Billing model id to which subscription belongs
  - The unique reference of subscription
  - The address of user who cancelled the subscription

getPullPayment()

Method name:

- getPullPayment

Method detail:

- This method retrieves the pullPayment details for given pullPayment id.
- It gives the payment amount, pullPayment timestamp, billing model id and the subscription id.
- Only the granted executor, payee and the subscriber of pullPayment can see the payment amount and execution timestamp.

Method signature:

- function getPullPayment(uint256 _pullPaymentID)
  external
  view
  returns (PullPaymentData memory pullPayment)

Method parameters:

- **_pullPaymentID:**
  - Indicates the valid pullPayment id.

Returned data:

- This method returns the pullPayment data for given pullPayment id which includes-
  - Payment amount
  - Timestamp when pullPayment executed.
  - Billing model to which this pullpayment belongs to
  - Subscription id to which this pullpayment belongs to

getBillingModelIdsByAddress()

Method name:

- getBillingModelIdsByAddress

Method detail:

- This method retrieves the billing model ids for a given creator address.

Method signature:

- function getBillingModelIdsByAddress(address _creator)
  external
  view
  returns (uint256[] memory billingModelIDs)

Method parameters:

- **_creator:**
  - Indicates the billing model creator address whose billing model id is to retrieve.

Returned data:

- This method returns the lists of billing model ids that a creator has created.

getSubscriptionIdsByAddress()

Method name:

- getSubscriptionIdsByAddress

Method detail:

- This method retrieves the subscription ids for a given subscriber address.

Method signature:

- function getSubscriptionIdsByAddress(address _subscriber)
  external
  view
  returns (uint256[] memory subscriptionIDs)

Method parameters:

- **_subscriber:**
  - Indicates the subscriber address whose subscription id is to retrieve.

Returned data:

- This method returns the lists of subscription ids that a subscriber has subscribed.

getCanceledSubscriptionIdsByAddress()

Method name:

- getCanceledSubscriptionIdsByAddress

Method detail:

- This method retrieves the subscription ids which the subscriber has cancelled.

Method signature:

- function getCanceledSubscriptionIdsByAddress(address _subscriber)
  external
  view
  returns (uint256[] memory subscriptionIDs)

Method parameters:

- **_subscriber:**
  - Indicates the subscriber address whose cancelled subscription ids to be retrieved.

Returned data:

- This method returns the list of subscription ids which the subscriber has cancelled.

getPullPaymentsIdsByAddress()

Method name:

- getPullPaymentsIdsByAddress

Method detail:

- This method retrieves the list of pullPayment ids for a given subscriber.

Method signature:

- function getPullPaymentsIdsByAddress(address _subscriber)
  external
  view
  returns (uint256[] memory pullPaymentIDs)

Method parameters:

- **_subscriber:**
  - Indicates the subscriber address whose pullPayment ids to be retrieved.

Returned data:

- This method returns a list of pullPayment ids for a given subscriber.

getCurrentBillingModelId()

Method name:

- getCurrentBillingModelId

Method detail:

- This method returns the current billing model id of the pullpayment

Method signature:

- function getCurrentBillingModelId() external view virtual returns (uint256);

Returned data:

- This method returns the current billing model id.

getCurrentSubscriptionId()

Method name:

- getCurrentSubscriptionId

Method detail:

- This method returns the current subscription id of the pullpayment

Method signature:

- function getCurrentSubscriptionId() external view virtual returns (uint256);

Returned data:

- This method returns the current subscription id.

getCurrentPullPaymentId()

Method name:

- getCurrentPullPaymentId

Method detail:

- This method returns the current id of the pullpayment

Method signature:

- function getCurrentPullPaymentId() external view virtual returns (uint256);

Returned data:

- This method returns the current pullpayment id.

Events

BillingModelCreated

Event Name:

- BillingModelCreated

Event detail:

- This event is emitted when the creator creates a new billing model using **createBillingModel()** method.

Event signature:

- ```
  event BillingModelCreated(
      uint256 indexed billingModelID,
      address indexed payee,
      uint8 indexed recurringPPType
  );
  ```

Event data:

- **billingModelID:**
  - Indicates the newly created billing model id.
- **Payee:**
  - Indicates the payee address who created the billing model.
- **recurringPPType:**
  - Indicates the type of recurring pullPayment i.e free trial, paid trial and normal recurring pullPayment.
  ```

NewSubscription

Event name:

- NewSubscription

Event detail:

- This event is emitted when a subscriber subscribes to the billing model using **subscribeBillingModel()** method.

Event signature:

```
event NewSubscription(
        uint256 indexed billingModelID,
        uint256 indexed subscriptionID,
        address payee,
        address payer
);
```

Event data:

- **billingModelID:**
  - Indicates the valid billing model id that subscriber has subscribed to.
- **subscriptionID:**
  - Indicates the subscription id when a subscriber subscribes to a given billing model.
- **Payee:**
  - Indicates the address of the merchant.
- **payer:**
  - Indicates the subscriber address who subscribed to the billing model.

PullPaymentExecuted

Event name:

- PullPaymentExecuted

Event detail:

- This event is emitted when the pullPayment is executed for a given subscription.
- This event is emitted inside the **executePullPayment()** method.

Event signature:

- event PullPaymentExecuted(
  uint256 indexed subscriptionID,
  uint256 indexed pullPaymentID,
  uint256 indexed billingModelID,
  address payee,
  address payer,
  uint256 executionFee,
  uint256 userAmount,
  uint256 receiverAmount
  );

Event data:

- **subscriptionID:**
  - Indicates the subscription id whose pullPayment is executed.
- **pullPaymentID:**
  - Indicates the id of executed pullPayment.
- **billingModelID:**
  - Indicates the billing model id for which pullPayment is executed.
- **payee:**
  - Indicates the merchant address
- **payer:**
  - Indicates the subscriber address who paid for the subscription.
- **executionFee:**
  - Indicates the execution fee charged in PMA token.
- **userAmount:**
  - Indicates the amount of payment tokens that subscriber paid for the subscription
- **receiverAmount:**
  - Indicates the amount of settlement tokens that merchant received.

SubscriptionCancelled

Event name:

- SubscriptionCancelled

Event detail:

- This event is emitted when a subscriber cancels the subscription using **cancelSubscription()** method or due to the low balance in subscriber's account.

Event signature:

```
event SubscriptionCancelled(
        uint256 indexed billingModelID,
        uint256 indexed subscriptionID,
        address payee,
        address payer
  );
```

Event data:

- **billingModelID:**
  - Indicates the billing model id to which cancelled subscription belongs.
- **subscriptionID:**
  - Indicates the cancelled subscription id.
- **payee:**
  - Indicates the merchant address
- **payer:**
  - Indicates the subscriber address whose subscription is cancelled.

BillingModelEdited

Event name:

- BillingModelEdited

Event detail:

- This event is emitted when the billing model is edited using the **editBillingModel()** method.

Event signature:

- ```
  event BillingModelEdited(
    uint256 indexed billingModelID,
    address indexed newPayee,
    address indexed oldPayee,
    string newMerchantName,
    string newMerchantUrl
  );
  ```

Event data:

- **billingModelID:**
  - Indicates the edited billing model id.
- **newPayee:**
  - Indicates the updated payee address.
- **oldPayee:**
  - Indicates the payee address who edited the billing model.
- **newMerchantName:**
  - Indicates the updated merchant`s name.
- **newMerchantUrl:**
  - Indicates the merchant`s new personal URL.
  ```

Flow of Execution:

1. First merchant creates the billing model with required configuration for his business. Merchant needs to specify the payee address and the type of the recurring pullPaymen, merchant name, unique reference for billing model and the merchant url. Rest of the details are provided dynamically through the code-snippet.
2. When a subscriber subscribes to the billing model, the remaining details of the billing model are provided dynamically through the code snippet. This detail includes the name of the billing model, settlement token address, payment amount, frequency of pullPayments, total number of pullPayments, trial period if required and the initial amount for trial period in case of paid trial recurring pullPayment.
3. Subscriber should approve the unlimited payment tokens to the Executor contract before subscribing to the billing model.
4. Executor is the contract which gets the tokens from the subscriber, swaps the payment token to the settlement token and then transfers the settlement tokens to the payee.
5. When a customer subscribes to the Normal type of recurring pullPayment, The first pullPayment is executed immediately and rest of the pullPayments are executed after a fixed interval of time.
6. When a customer subscribes to the Free trial type of recurring pullPayment, The customer's free trial gets started. No amount is charged to the customer during the free trial period. After the completion of a free trial, the user is charged for the services via pullPayments.
7. When a customer subscribes to the Paid trial type of recurring pullPayment, The initial amount is charged to the customer and paid trial is activated. After the completion of the paid trial, the customer is charged for the services using the pullPayments.
8. The pullPayments are executed until the subscriber cancels the subscription or the total number of pullPayments gets completed.
9. Users can cancel the subscription if they don't want to continue the service offered by the merchant.
10. Merchants can edit the billing model name and payee address if they want.

Flow of Execution with Chainlink Keepers:

1. First merchant creates the billing model with required configuration for his business.
2. Subscriber approves the enough payment tokens to Executor contract if not approved already.
3. Subscriber subscribes the billing model with a specific recurring pullpayment type.
4. When a customer subscribes to the Normal type of recurring pullPayment, The first pullPayment is executed immediately and rest of the pullPayments are executed after a fixed interval of time.
5. When a customer subscribes to the Free trial type of recurring pullPayment, The customer's free trial gets started. No amount is charged to the customer during the free trial period. After the completion of a free trial, the user is charged for the services via pullPayments.
6. When a customer subscribes to the Paid trial type of recurring pullPayment, The initial amount is charged to the customer and paid trial is activated. After the completion of the paid trial, the customer is charged for the services using the pullPayments.
7. Keeper Nodes continuously calls the checkUpkeep() method to know whether there are pull payments to execute or not.
8. If there are subscriptions in the list returned by checkUpkeep(), Keeper node calls the performUpkeep() method with the performData returned by checkUpkeep().
9. Perform upkeep executes pull payment for each of the subscriptions specified in the list.
10. If any subscriber does not have the enough amount of payment tokens in his wallet, then that subscription is added to the low balance subscriptions list
11. This low balance subscription is again considered for the execution only after the extension period.
12. After the extension period, if the subscriber still does not have enough tokens then it cancels the subscription.
13. If a subscriber has added the tokens to his account during the extension period then the subscription is removed from the low balance subscriptions list and pull payment is executed normally.