Описание SDK Ruby

Содержание

Введение

Документ описывает Software Development Kit (SDK) на Ruby.

Документ адресован разработчикам, имеющим базовые знания языка Ruby, подключающим оплату через систему «MONETA.RU».

Глава 1

Скачивание и установка

SDK Ruby состоит из двух составляющих, которые находятся в репозиториях.

Payanyway

Этот предназначен для быстрой интеграции платежного шлюза payanyway в ваше ruby приложение.

Для установки, добавьте эти строки в Gemfile вашего приложения:

```
gem 'payanyway'
```

И выполните команду:

```
bundle
```

Или установите gem напрямую через командную строку Ruby:

```
gem install payanyway
```

Merchant API

Добавьте эти строки в Gemfile вашего приложения:

```
gem 'moneta-api'
```

И выполните команду:

```
bundle
```

Или установите данный gem напрямую через командную строку Ruby:

```
gem install moneta-api
```

Глава 2

Основные настройки

Для быстрой интеграции платежного шлюза создайте конфигурационный файл config/payanyway.yml:

```
development: &config
    moneta_id: YOUR_MONETA_ID
    currency: RUB
    payment_url: https://demo.moneta.ru/assistant.htm
    test_mode: 1
    token: secret_token
production: <<: *config
    payment_url: https://moneta.ru/assistant.htm
    test_mode: 0
```

Для работы с Merchant API надо настроить Logger:

```
Moneta::Api::Service.new('username', 'password', {
    logger: Rails.logger,
    log_level: :info,
    log: true,
    filters: [:password]
})
```

Подготовка к работе

Добавьте engine в файл config/routes.rb:

```
Rails.application.routes.draw do
  mount Payanyway::Engine => '/payanyway'
end
```

Создайте файл app/controllers/payanyway_controller.rb со следующим кодом:

```
class PayanywayController < ApplicationController
  include Payanyway::Controller

  def success_implementation(transaction_id)
    # вызывается при отправке шлюзом пользователя на Success URL.
    #
    # ВНИМАНИЕ: является незащищенным действием!
    # Для выполнения действий после успешной оплаты используйте pay_implementation
  end

  def pay_implementation(params)
    # вызывается при оповещении магазина об
    # успешной оплате пользователем заказа. (Pay URL)
    #
    # params[ KEY ], где KEY # [ :moneta_id, :transaction_id, :operation_id,
    # :amount, :currency, :subscriber_id, :test_mode, :user, :corraccount,
    # :custom1, :custom2, :custom3 ]
  end

  def fail_implementation(transaction_id)
```

```
    # вызывается при отправке шлюзом пользователя на Fail URL.
  end
end
```

Для локального запуска тестов необходимо создать конфигурационный файл со своими demo доступами в Merchant API:

```
echo "username: 'username'\npassword: 'password'" > spec/support/moneta.yml
```

Глава 3

Методы SDK

Запрос на оплату

Чтобы получить ссылку на платежный шлюз для оплаты заказа пользователем, используйте метод:

```
Payanyway::Gateway.payment_url(params, use_signature = true)
```

Где **params** - это массив с параметрами, имеющий следующие ключи:

:transaction_id - идентификатор операции внутри интернет-магазина,

:amount - сумма операции,

:test_mode - значение 0 или 1, если тестовый режим включен,

:description - описание операции,

:subscriber_id - идентификатор покупателя в интернет-магазине,

:custom1 - доп. параметр 1,

:custom2 - доп. параметр 2,

:custom3 - доп. параметр 3,

:locale - язык пользовательского интерфейса (ru или en),

:payment_system_unit_id - выбранный способ оплаты,

:payment_system_limit_ids - строка с идентификаторами способов оплаты, доступными для выбора в платежной форме (указываются через запятую),

:success_url - URL страницы магазина, куда должен попасть покупатель после успешной оплаты,

:in_progress_url - URL страницы магазина, куда должен попасть покупатель после успешного запроса на авторизацию средств,

:fail_url - URL страницы магазина, куда должен попасть покупатель после отмененной или неуспешной оплаты,

:return_url - URL страницы магазина, куда должен вернуться покупатель при добровольном отказе от оплаты.

Пример минимальной ссылки:

```
class Order < ActiveRecord::Base; end

class OrdersController < ApplicationController
  def create
    order = Order.create(params[:order])
    redirect_to Payanyway::Gateway.payment_url(
```

```
      transaction_id: order.id,
      amount: order.total_amount,
      description: "Оплата заказа № #{ order.number } на сумму
 #{ order.total_amount }руб."
    )
  end
end
```

При необходимости можно переопределить moneta_id, currency, test_mode, так же передав их в payment_url.

Обработка уведомлений от Moneta.ru

Система Moneta.ru отправляет уведомления Pay URL при зачислении платежа и Check URL перед проведением платежа.

Gem payanyway добавляет специальные роуты для обработки этиъ запросов от шлюза.

Check URL:

```
class PayanywayController
  ...
  def check_implementation(params)
    # Вызывается при обработке проверочных запросов (Check URL)
    # params[ KEY ], где KEY - [ :moneta_id, :transaction_id, :operation_id,
    # :amount, :currency, :subscriber_id, :test_mode, :user, :corraccount,
    # :custom1, :custom2, :custom3, :payment_system_unit_id ]

    # ВНИМАНИЕ: при отправке корректного ответа со стороны магазина,
    #   необходимо вернуть в методе параметры для генерации статус-кода.
    #   { amount: AMOUNT, state: STATE, description: DESCRIPTION,
    #   attributes: ATTRIBUTES, logger: true\false }
  end
end
```

Пример кода:

```
...
def check_implementation(params)
  order = Order.find(params[:transaction_id])
  {
    amount: order.total_amount,
    state: order.state_for_payanyway, # нужно реализовать
    attributes: { name: 'John Smith', email: 'js@gmail.com' }
  }
end
...
```

Возвращаемые параметры:

Название	Описание
:amount	Сумма заказа.
:state	Состояние оплаты заказа.
:description	Описание состояния заказа. Задается в произвольной форме.
:attributes	Необязательный элемент. Содержит хеш произвольных параметров, которые будут сохранены в операции.
:logger	Вывести XML ответ в log (Rails.logger).

Возможные состояния оплаты заказа:

Состояние	Описание
:paid	Заказ оплачен. Уведомление об оплате магазину доставлено.

Состояние	Описание
:in_progress	Заказ находится в обработке. Точный статус оплаты заказа определить невозможно. (например, если пользователя отправило на InProgress URL, но уведомления на Pay URL от шлюза еще не поступало).
:unpaid	Заказ создан и готов к оплате. Уведомление об оплате магазину не доставлено.
:canceled	Заказ не является актуальным в магазине (например, заказ отменен).

Return URL и InProgress URL:

```
class PayanywayController
  ...
  def return_implementation(transaction_id)
    # Вызывается при добровольном отказе пользователем от оплаты (Return URL)
  end

  def in_progress_implementation(transaction_id)
    # Вызывается после успешного запроса на авторизацию средств,
    # до подтверждения списания и зачисления средств (InProgress URL)
    #
    # ВНИМАНИЕ: InProgress URL может быть использован в любом способе оплаты.
    #   Если к моменту, когда пользователя надо вернуть в магазин оплата,
    #   по какой-либо причине не завершена, то его перекинет на InProgress,
    #   если он указан, если не указан, то на Success URL.
    #   Если операция уже успешно выполнилась, то сразу на Success.
    #
  В случае с картами чаще всего получается так, что операция не успевает выполниться,
    #   поэтому InProgress будет использован с бОльшей вероятностью, чем Success URL.
  end
  ...
end
```

Расшифровка параметров:

params[KEY], где KEY ...	В документации	Описание
:moneta_id	MNT_ID	Идентификатор магазина в системе MONETA.RU.
:transaction_id	MNT_TRANSACTION_ID	Внутренний идентификатор заказа, однозначно определяющий заказ в магазине.
:operation_id	MNT_OPERATION_ID	Номер операции в системе MONETA.RU.
:amount	MNT_AMOUNT	Фактическая сумма, полученная на оплату заказа.
:currency	MNT_CURRENCY_CODE	ISO код валюты, в которой произведена оплата заказа в магазине.
:test_mode	MNT_TEST_MODE	Флаг оплаты в тестовом режиме (1 - да, 0 - нет).
:description	MNT_DESCRIPTION	Описание оплаты.
:subscriber_id	MNT_SUBSCRIBER_ID	Внутренний идентификатор пользователя в системе магазина.
:corraccount	MNT_CORRACCOUNT	Номер счета плательщика.
:custom[1|2|3]	MNT_CUSTOM1	Поля произвольных параметров. Будут возвращены магазину в параметрах отчета о проведенной оплате.
:user	MNT_USER	Номер счета пользователя, если оплата производилась с пользовательского счета в системе «MONETA.RU».
:locale	moneta.locale	(ru|en) Язык пользовательского интерфейса.

params[KEY], где KEY ...	В документации	Описание
:success_url	MNT_SUCCESS_URL	URL страницы магазина, куда должен попасть покупатель после успешно выполненных действий.
:in_progress_url	MNT_INPROGRESS_URL	URL страницы магазина, куда должен попасть покупатель после успешного запроса на авторизацию средств, до подтверждения списания и зачисления средств.
:fail_url	MNT_FAIL_URL	URL страницы магазина, куда должен попасть покупатель после отмененной или неуспешной оплаты.
:return_url	MNT_RETURN_URL	URL страницы магазина, куда должен вернуться покупатель при добровольном отказе от оплаты.
:attributes	MNT_ATTRIBUTES	Содержит произвольные параметры, которые будут сохранены в операции.

Параметры, отвечающие за выбор платежной системы:

params[KEY], где KEY ...	В документации	Описание
:payment_system_unit_id	paymentSystem.unitId	Выбранная платежная система.
:payment_system_limit_ids	paymentSystem.limitIds	Список (разделенный запятыми) идентификаторов платежных систем для выбора в форме оплаты.

Методы Merchant API

Методы Merchant API позволяет организовать взаимодействие между системой Moneta.ru и внешней системой.

Вызов методов Merchant API. Базовый пример:

```
require 'moneta/api'

  # получить данные счета
  service = Moneta::Api::Service.new('username', 'password', { demo_mode: true })
  response = service.find_account_by_id(10999)
  puts response.class.name
  # => 'Moneta::Api::Responses::FindAccountByIdResponse'
  puts "Баланс до пополнения: #{ response.account.balance}
 #{ response.account.currency }"
  # => 'Баланс до пополнения: 100 RUB'

  # перевод
  transfer_request = Moneta::Api::Requests::TransferRequest.new.tap do |request|
    request.payee = 28988504
    request.payer = 10999
    request.amount = 10
    request.is_payer_amount = false
    request.payment_password = '123456'
  end
  response = service.transfer(transfer_request)

  # данные транзакции
  puts response.class.name
  # => 'Moneta::Api::Responses::TransferResponse'
  puts response.status
  # => 'SUCCEED'

  # проверить данные счета
  response = service.find_account_by_id(10999)
```

```
  puts "Баланс после пополнения: #{ response.account.balance}
#{ response.account.currency }"
  # => 'Баланс после пополнения: 110 RUB'
```

Полный список методов Merchant API, поддерживаемых данным SDK можно увидеть по ссылке: *http://www.rubydoc.info/gems/moneta-api/Moneta/Api/ServiceMethods*

Для использования тестового сервера (http://demo.moneta.ru) следует инициализировать сервис со специальным флагом:

```
Moneta::Api::Service.new('username', 'password', { demo_mode: true })
```

Chapter 4

Полезные ссылки

Topics:

SDK на различных языках

- *C#*
- *PHP*
- *Python3*
- *Ruby*
- *Java*

Документация по MONETA.Assistant

- *https://www.moneta.ru/doc/MONETA.Assistant.ru.pdf*

Описание методов MONETA.MerchantAPI

- *https://www.moneta.ru/doc/MONETA.MerchantAPI.v2.ru.pdf*