WayForPay

**WayForPay** — це український платіжний сервіс, який дозволяє приймати онлайн-оплати з банківських карт, Apple Pay і тд. Його часто використовують інтернет-магазини та сервіси для швидкого підключення платіжної системи без зайвих формальностей.

Документація API:

* [Загальна документація](https://wiki.wayforpay.com/) [1]
* [Створення рахунку](https://wiki.wayforpay.com/view/608996852) [2]
* [Тестові реквізити](https://wiki.wayforpay.com/view/852472) [3]

Основне завдання, яке буде стояти перед розробником – правильно створити invoice, аби перенаправити користувача на сторінку з оплатою, та правильно обробити відповідь від WayForPay. Отже перейдемо відразу до діла.

# **Створення invoice**

Для створення invoice для початку нам потрібні будуть реквізити на кого він буде створюватись. На щастя, сервіс надає тестові реквізити[3], для тесту нам знадобляться лиш поля з “Прийняти платіж” – merchantAccount та merchantSecretKey.

На жаль, у WayForPay немає своєї бібліотеки як у LiqPay, яка б спростила наше завдання, тому доведеться розбиратися з запитами самим. Отож, щоб створити invoice, якщо слідувати документації[2], то все що нам потрібно зробити, це зробити звичайний POST запит до їхнього сервісу. Однак за цим ховається головна складність – створення сигнатури. Сигнатура – це такий собі підпис, який використовується для перевірки автентичності даних. Тобто, якщо відправлений нами і згенерований сервісом підписи не співпадають, то ми отримаємо помилку оплати. Гайд по тому як створити сигнатуру написаний у самому верху сторінки зі створенням рахунку[2]. На щастя, я можу надати приклад функції, яка набагато спростить роботу (Код 1). Ця функція є більш точковою, адже розроблялася для своїх цілей. Як можна побачити, я отримую параметр data та дістаю з нього потрібні дані, які записую в окрему строку, де кожен параметр розділено “;”. Прошу зауважити, що в сигнатурі також використовується self.account та self.domain. Цей приклад коду вирізаний з спеціально написаного для роботи з WayForPay класу, код якого я надаю нижче (в кінці файлу). Наш account це назва акаунту рахунку, domain – посилання, на яке кине відповідь сервіс. Також прошу звернути увагу, що не всі поля, зазначені у документації, є обов’язковими: у вас їх може бути менше, а може й більше. Головне – сигнатура повинна містити поля, які позначені обов’язковими у документації сервісу. Також зверніть увагу на те як шифрується сигнатура – за допомогою self.keyце наш секретний ключ, який можна отримати на сторінці із тестовими реквізитами[3].

def get\_signature(self, data):

order\_reference = data['orderReference']

order\_date = data['orderDate']

amount = data['amount']

currency = data['currency']

product\_names: list = data['productName']

product\_counts: list = data['productCount']

product\_prices: list = data['productPrice']

signature\_text = (

f"{self.account};"

f"{self.domain};"

f"{order\_reference};"

f"{order\_date};"

f"{amount};"

f"{currency};"

f"{";".join(product\_names)};"

f"{";".join(str(count) for count in product\_counts)};"

f"{";".join(str(price) for price in product\_prices)}"

)

signature = hmac.new(self.key.encode("utf-8"), signature\_text.encode("utf-8"), hashlib.md5).hexdigest()

return signature

Код 1 – Приклад створення сигнатури

Гаразд, із найскладнішим розібралися. Перейдемо до створення самого invoice. Маючи сигнатуру, все, що нам лишається зробити, це надіслати сам запит. Відповідний код надаю:

def create\_invoice(self, data):

try:

body = {

"merchantSignature": self.get\_signature(data),

"merchantAccount": self.account,

"merchantDomainName": self.domain,

"transactionType": "CREATE\_INVOICE",

"apiVersion": "1",

"language": "ua",

"notifyMethod": "email"

}

body.update(data)

headers = {"Content-Type": "application/json"}

response = requests.post(self.API\_URL, json=body, headers=headers)

if response.text == "Api cannot read incoming request data. Unknown format":

raise ValueError("Unknown format")

return response.json()

except KeyError as e:

raise KeyError(f"No required argument – {e}")

except Exception as e:

raise e

Код 2 – Приклад створення invoice

Тут ми знову ж передаємо всі необхідні дані (body.update(data)), вказуємо мову, яку хочемо бачити на сторінці з оплатою, та решту полів. Усе що лишається – прописати headers та відправити запит!

При успішному запиті у відповідь нам прийде словник із полем invoiceUrl, на яке нам потрібно буде перенаправити користувача. Надаю приклад використання:

wfp = WayForPay(os.getenv("MERCHANT\_ACCOUNT"), os.getenv("MERCHANT\_SECRET\_KEY"), os.getenv("MERCHANT\_DOMAIN"))

invoice = wfp.create\_invoice({

"orderReference": f"order\_{order\_id}",

"orderDate": int(now.timestamp()),

"orderTimeout": 900, #15 хв на оплату

"amount": amount,

"currency": "UAH",

"productName": product\_name,

"productCount": product\_count,

"productPrice": product\_price,

"clientFirstName": name,

"clientLastName": surname,

"clientEmail": email,

"clientPhone": phone,

"serviceUrl": f"{os.getenv("MERCHANT\_DOMAIN")}accept\_payment",

"returnUrl": f"{os.getenv("MERCHANT\_DOMAIN")}order\_status/{order\_id}/{email}"

})

#Пізніше у коді

return jsonify(invoice.get("invoiceUrl")) # Перенаправить користувача на сторінку оплати

Код 3 – Приклад створення invoice на практиці

# **Отримання відповіді**

Після оплати користувачем сервіс зобов'язаний сповістити нас про це за merchantDomainName, яке ми надали. Прийнявши цю відповідь ми можемо отримати дані про те як пройшов платіж та записати необхідну інформацію у базу даних, наприклад. Однак на цей Callback ми зобов'язані відповісти, інакше сервіс надішле ще декілька таких ж. Стається це через те, що WayForPay не розуміє чи успішно до нас дійшло його повідомлення і чи обробили ми його. Можливе усяке, наприклад, що наш сервер у той чи інший момент лежав, через що ми не змогли прийняти відповідь, тощо. Отож, щоб дати знати, що у нас все добре, формуємо відповідь:

def get\_answer\_signature(merchant\_key, data):

order\_reference = data["orderReference"]

status = data["status"]

time = data["time"]

signature\_text = (

f"{order\_reference};"

f"{status};"

f"{time}"

)

signature = hmac.new(merchant\_key.encode("utf-8"), signature\_text.encode("utf-8"), hashlib.md5).hexdigest()

return signature

Код 4 – Приклад створення відповіді

На щастя, даних тут набагато менше. Із важливого: time повинно бути int інакше відповідь не буде прийнята (так, надіслати у відповідь сервісу невідомо що також не підходить).

Пропоную відразу розглянути наглядний приклад використання цієї функції:

@routes.route("/accept\_payment", methods=['POST'])

def accept\_payment():

try:

raw\_data = next(iter(request.values.keys()), '{}')

data = json.loads(raw\_data)

now = datetime.now()

if data["reasonCode"] == 1100: # Обробка успішної відповіді

else: # Обробка неуспішної відповіді

answer = {

"orderReference": data["orderReference"],

"status": "accept",

"time": int(now.timestamp()),

}

answer["signature"] = WayForPay.get\_answer\_signature(os.getenv("MERCHANT\_SECRET\_KEY"), answer)

return jsonify(answer), 200

except Exception as e:

return jsonify({"error": str(e)}), 500

Код 5 – Приклад відповіді на практиці

Отож, ми розібрали усе необхідне для базового розуміння роботи із сервісом WayForPay. Успіхів вам у коді!

# **Практичне завдання**

Для реалізації завдання дозволяється використовувати будь-який зручний спосіб створення ендпоінтів.

* [Завдання](https://docs.google.com/document/d/1r20kACEvYra8ugo_K8h42i1CdfrszVZjgiXjrmzCbQ8/edit?tab=t.0)

# **Клас для роботи з WayForPay**

import hashlib

import hmac

import requests

class WayForPay:

API\_URL = "https://api.wayforpay.com/api"

def \_\_init\_\_(self, account, key, domain):

self.account = account

self.key = key

self.domain = domain

@staticmethod

def get\_answer\_signature(merchant\_key, data):

order\_reference = data["orderReference"]

status = data["status"]

time = data["time"]

signature\_text = (

f"{order\_reference};"

f"{status};"

f"{time}"

)

signature = hmac.new(merchant\_key.encode("utf-8"), signature\_text.encode("utf-8"), hashlib.md5).hexdigest()

return signature

def get\_signature(self, data):

order\_reference = data['orderReference']

order\_date = data['orderDate']

amount = data['amount']

currency = data['currency']

product\_names: list = data['productName']

product\_counts: list = data['productCount']

product\_prices: list = data['productPrice']

signature\_text = (

f"{self.account};"

f"{self.domain};"

f"{order\_reference};"

f"{order\_date};"

f"{amount};"

f"{currency};"

f"{";".join(product\_names)};"

f"{";".join(str(count) for count in product\_counts)};"

f"{";".join(str(price) for price in product\_prices)}"

)

signature = hmac.new(self.key.encode("utf-8"), signature\_text.encode("utf-8"), hashlib.md5).hexdigest()

return signature

def create\_invoice(self, data):

try:

body = {

"merchantSignature": self.get\_signature(data),

"merchantAccount": self.account,

"merchantDomainName": self.domain,

"transactionType": "CREATE\_INVOICE",

"apiVersion": "1",

"language": "ua",

"notifyMethod": "email"

}

body.update(data)

headers = {"Content-Type": "application/json"}

response = requests.post(self.API\_URL, json=body, headers=headers)

if response.text == "Api cannot read incoming request data. Unknown format":

raise ValueError("Unknown format")

return response.json()

except KeyError as e:

raise KeyError(f"No required argument – {e}")

except Exception as e:

raise e