



Vipps Developer Portal

User Manual v1.0

About this Manual

This manual aims to give guidance in Vipps Developer Portal and describe features and functionalities available.

Purpose

This manual presents the functional capabilities and operational details of the developer portal and explains the procedures for utilizing the features and functionalities available in the portal.

Who should read this manual?

This manual is intended for those who will be using the developer portal to register and configure their applications that interact with Vipps APIs.

Table of Contents

1. Introduction.....	4
2. Signing in to the Portal.....	5
3. Features and Functionalities	6
3.1 User Profile	6
3.2 Products.....	7
3.3 Product Details.....	8
3.4 Register an Application	11
3.5 View registered applications.....	13
3.5.1 Regenerate Client secret.....	13
3.5.2 Remove a Registered Application	14
3.6 How to Test the APIs	15
3.6.1 Get an Access Token	15
3.7 Reports & Analytics	20
4. How to call Vipps APIs from application.....	20
4.1 Define an Application Cache	20
2.2 Get Access Token.....	21
2.3 Call an API	22

1. Introduction

This manual describes features and operational details of Vipps Developer Portal and explains the procedures for utilizing the features and functionalities available in the portal.

Vipps developer portal serves as the main web presence for developers, where user can:

- Read API documentation.
- Try out an API via the interactive console.
- View analytics on their usage.

What is available in the portal?

Vipps developer portal facilitates merchant developers to perform following things:

- Test the APIs via the interactive console
- Register an application to get API access
- View registered applications
- Remove a registered application
- View user profile information
- View analytics and usage reports

All the above features and functionalities are described in detail on the later sections.

2. Signing in to the Portal

To use Vipps Developer Portal user must go to the <https://api-portal.vipps.no> (Production) or <https://apitest-portal.vipps.no> (Test) URL which will land user onto the home page of the portal. The home page looks like below.

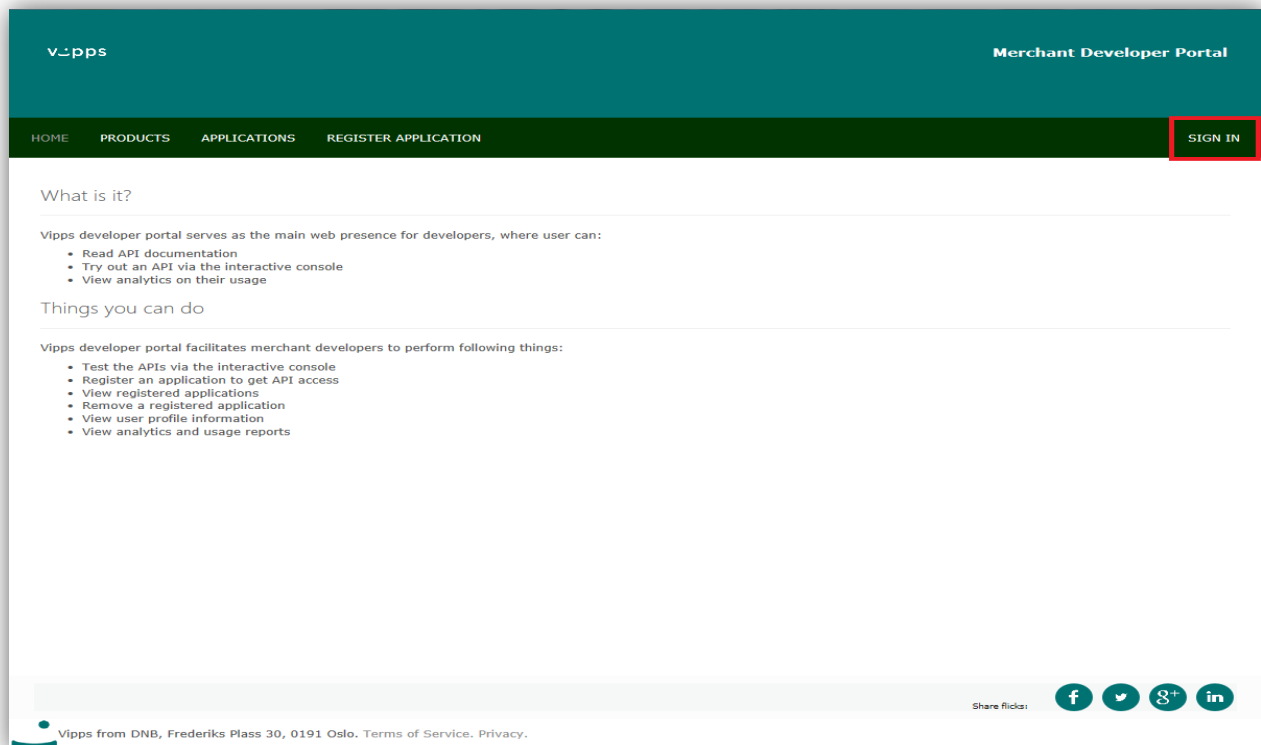


Image 1: Home Page

User must sign into the portal before using any functionality. To sign in, please click on the Sign In link at the top right corner of the page. It will display the sign in option. Click "**Azure Active Directory**" link and user will be redirected to the sign in page as shown below.

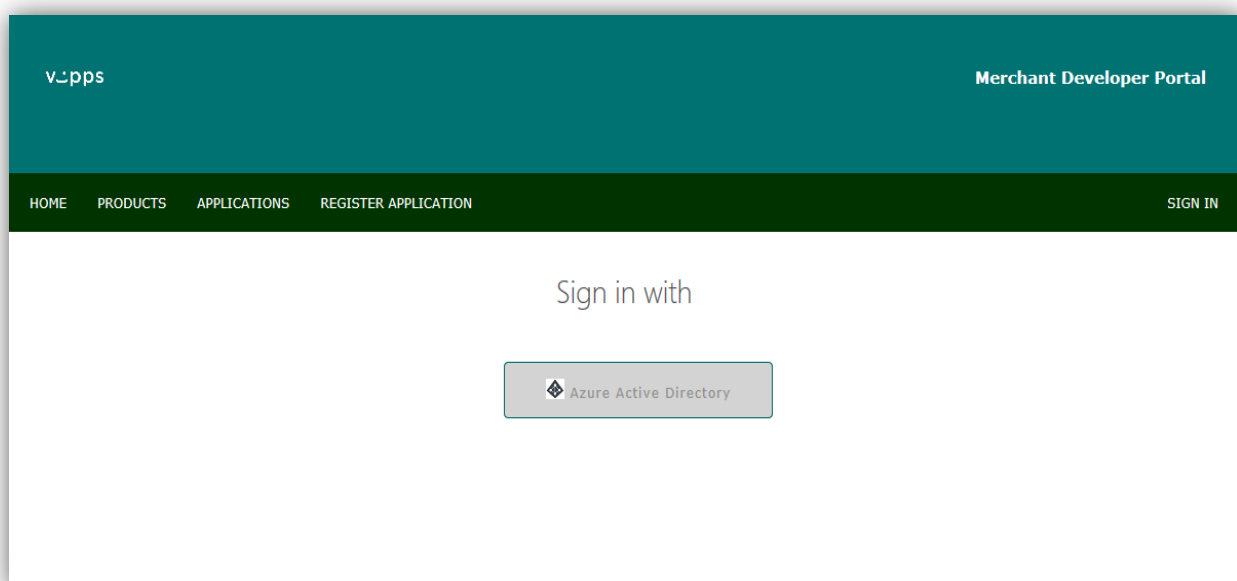


Image 2: Sign-in Option

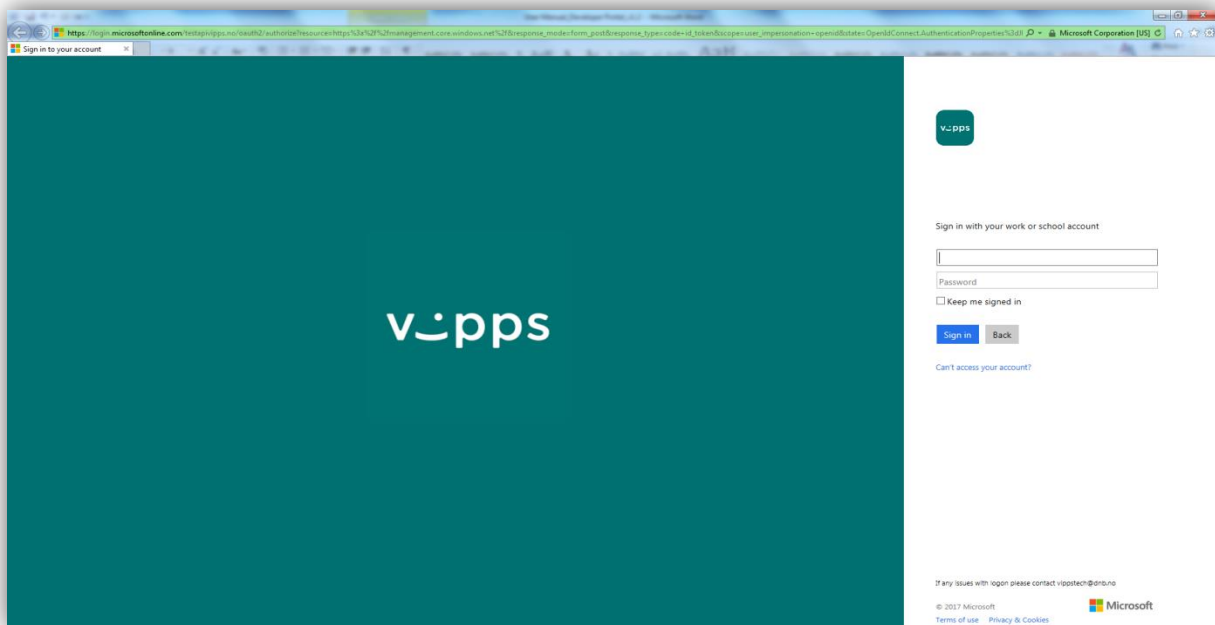


Image 3: Sign-in Page

Enter the username sent in email and password received by SMS to logon to the portal. User needs to change the default password on first login. On successful login user will be redirected back to home page where he can see the login name is displayed at the top right corner of the page. Now user can use the functionality available in the portal.

3. Features and Functionalities

The following sections describe the various functionalities a merchant developer should learn before they can successfully interact with Vipps APIs.

3.1 User Profile

User can view his/her Profile details and Subscription details by clicking on the profile link under the signed in name on the top right corner.

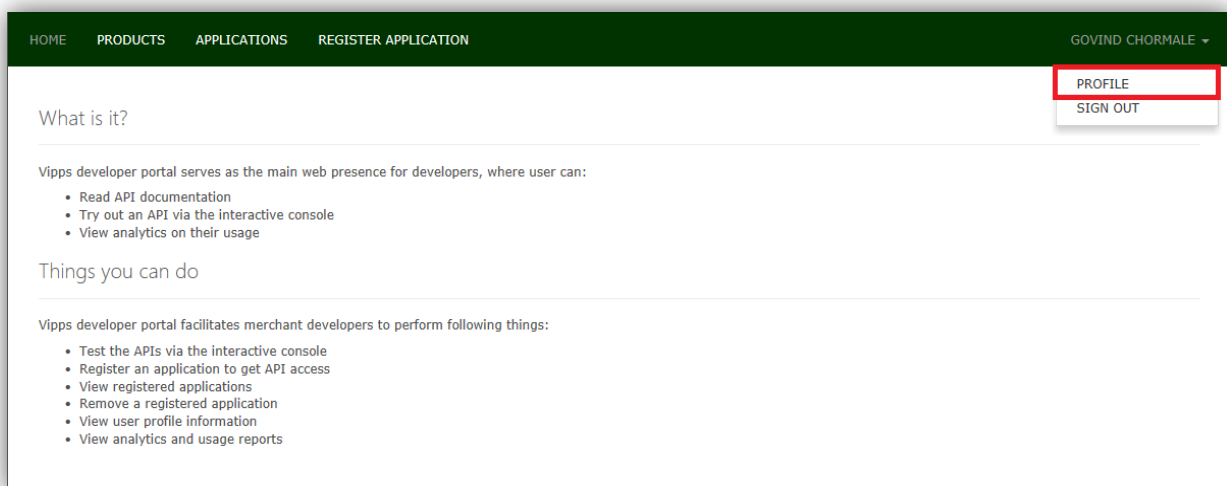


Image 4: User profile link

Profile:

Profile contains the information of user signed in. Basic information like Email, first name and last name of the user will be displayed here.

Subscriptions:

This contains all the subscribed products details which include Subscription name, primary key, secondary key(these are subscription key), product name and state of that product. The primary and secondary keys will be hidden by default. The key can be seen by clicking on show button corresponding to the product and key. The keys can be regenerated by clicking regenerate link corresponding to each link.

The screenshot displays the user profile and subscription management interface. The top navigation bar includes links for HOME, PRODUCTS, APPLICATIONS, and REGISTER APPLICATION, along with the user's name GOVIND CHORMALE. The profile section shows the user's email (Govind.chormale@dnb.no), first name (Govind), and last name (Chormale). Below this, the 'Your subscriptions' section features an 'Analytics reports' button and a table of active subscriptions.

Subscription details		Product	State
Subscription name	DEFAULT_ACCESSTOKEN	Access Token	Active
Primary key	[Redacted]	Rename	Show Regenerate
Secondary key	[Redacted]	Show Regenerate	Show Regenerate
Subscription name	ECOMMERCE-Test 100300	eCommerce	Active
Primary key	[Redacted]	Rename	Show Regenerate
Secondary key	[Redacted]	Show Regenerate	Show Regenerate
Subscription name	INAPP-Inapp test 100101	InApp	Active
Primary key	[Redacted]	Rename	Show Regenerate
Secondary key	[Redacted]	Show Regenerate	Show Regenerate
Subscription name	INVOICE-Invoice APIM 100103	Invoice	Active
Primary key	[Redacted]	Rename	Show Regenerate
Secondary key	[Redacted]	Show Regenerate	Show Regenerate

Image 5: View profile

Note: The goal of the primary and secondary keys is to allow for "rolling" upgrades, merchant can have a client with both, if one isn't working the other key can be used. This allows for the scenario to change the primary key (regenerate) and the other client can still use the secondary keys. After the primary is regenerated, the same scenario can be followed with the secondary key to regenerate that.

3.2 Products

Products are equal to the products that are agreed on in the commercial agreement. Merchant will only have access to the relevant products that are subscribed when merchant is on boarded into Vipps system. Each product is an accumulation of Vipps APIs that are exposed to merchants. All the Vipps APIs require access token to be passed in every request. So every merchant is subscribed to Access Token product by default. The API in Access Token product helps to get the access token.

Merchant can see the subscribed products in the portal by clicking on the products tab as shown below. If same merchant subscribe multiple products, he can see all the products here, as shown for this sample user. "Access Token" product is default for any merchant. Every product item in the list gives basic details of that particular product. To see more details and test the API services user needs to click on a product.

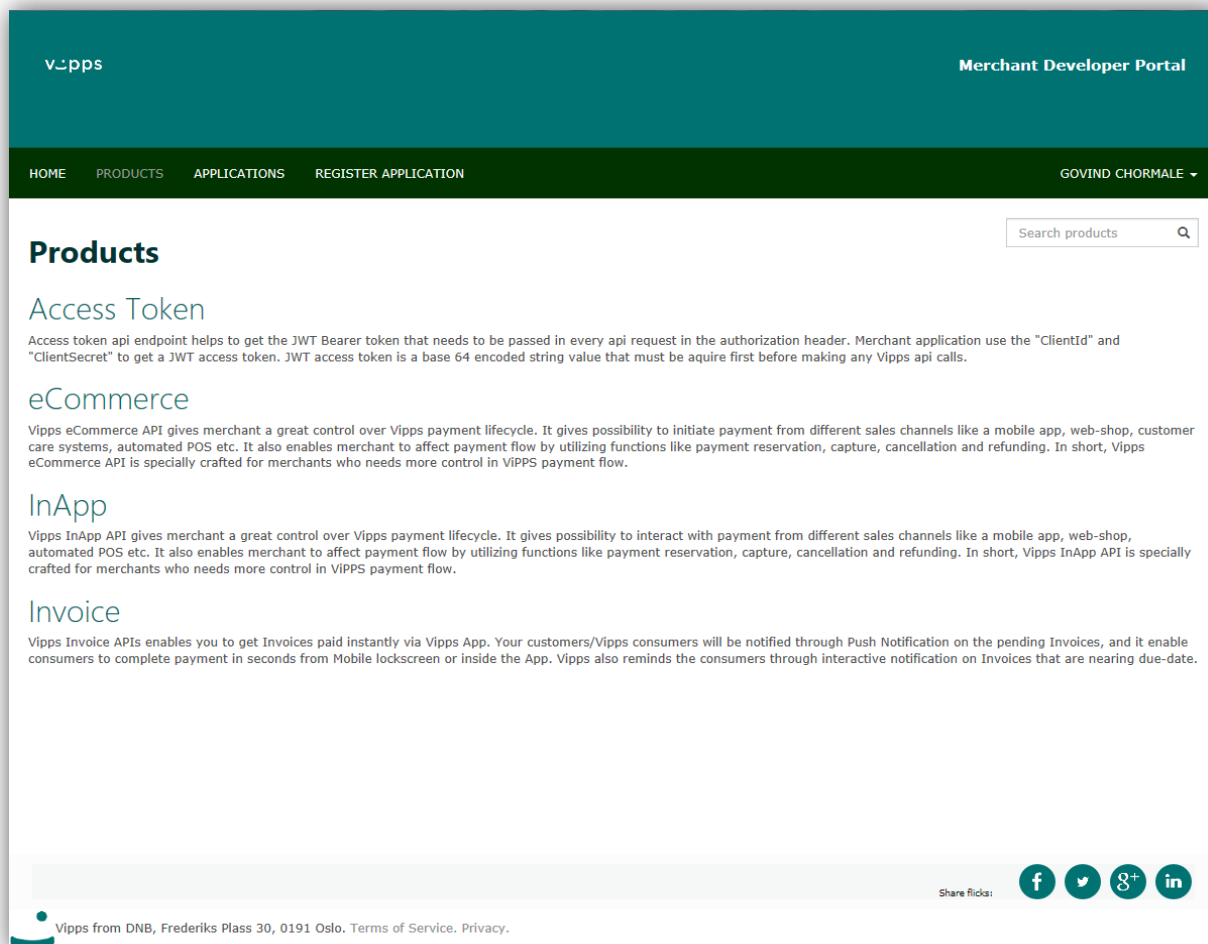


Image 6: Product List

3.3 Product Details

Each Product represents a set of API operations application can invoke to consume a backend resource. Each API operation contains a reference to the back-end service that implements the API and its operations map to the methods implemented by the back-end service.

User can see all the API services of each product and test the same by clicking on the **"TEST THE API(S)"** button.

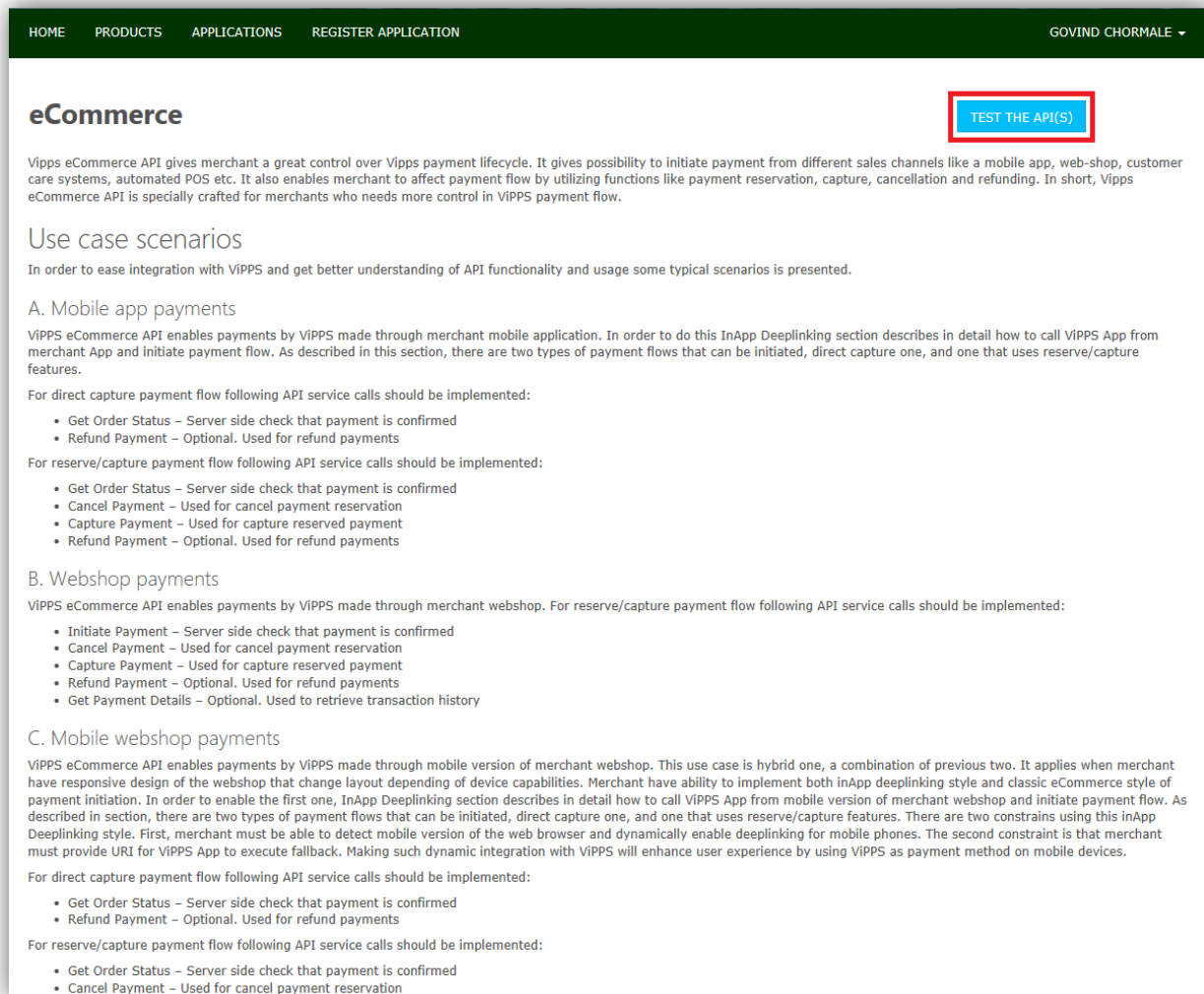


Image 7: Product Details

Clicking on the “TEST THE API(S)” button will take user to a page where user can see all the details of the APIs given below.

- List of operations in the API with HTTP verb type like GET, POST etc.
- Description of the API and it’s all Operations.
- Request and Response format.
- Code samples in various languages for how to invoke an operation.
- User can download API definition by clicking on **API Definition** button on the page.

Cancel Payment

The API call allows merchant to cancel the reserved transaction, The API will not allow partial cancellation which has a consequence that partially captured transactions cannot be cancelled.

Please note that in a case of communication errors during initiate payment service call between ViPPS and PSP/Acquirer/Issuer; even in a case that customer has confirmed a payment, the payment will be cancelled by ViPPS.

[Try it](#)

Request URL

https://apitest.vipps.no/Ecomm/v1/payments/{orderId}/cancel

Request parameters

Parameter	Type	Description
orderId	string	Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters

Request headers

Header	Type	Description
Authorization	string	Authorization token, is obtained by registering merchant backend application in Merchant Developer Portal.
X-Request-Id	string	For Making request to be idempotent this ID is must so that the system will not do any side effects. 1. Mandatory for Initiate,Capture, Refund payment 2. Size should be 30 3. Optional for Cancel Payment, Get Payment Details and Get Order Status. 4. If user wants to re-try any failed capture or refund transaction then they should provide same X-request-id, else system will create a new entry for partial capture or partial refund.
X-TimeStamp	string	Time stamp when the request called
X-Source-Address	string	Either source ip address or device id and terminal id for mobile application. This is for Identifying the request source
X-App-Id	string	Client Id generated while registering the application in Merchant developer portal for eCommerce product.
Content-Type (optional)	string	Media type of the body sent to the API.

Image 8: APIs in a product

Request body

"merchantSerialNumber" is a 6 characters long string parameter that identifies a merchant sales channel i.e. website, mobile app etc. This is required.

"transactionText" is a max 100 characters long string parameter for Reference text for the merchant. This is required.

application/json

```
{
  "merchantInfo": {
    "merchantSerialNumber": "NSBWSHP12"
  },
  "transaction": {
    "transactionText": "transaction text"
  }
}
```

Image 9: Request Body of a sample API

Response 200

"orderId" uniquely identifies a payment. Maximum length is 30 alphanumeric characters

"amount" is an Integer value in øre

"timeStamp" is a String in ISO-8601 representing when vipps Cancelled transaction.

"transactionText" is a String, transaction text reference provided by merchant.

"status" is a String defines the status of the ordered transaction.

"transactionId" is a String of ViPPS transaction id

"capturedAmount" is an Integer representing total amount captured.

"remainingAmountToCapture" is an Integer representing total remaining amount to capture.

"refundedAmount" is an Integer representing total refunded amount of the order.

"remainingAmountToRefund" is an Integer representing total remaining amount to refund.

application/json */*

```
{
  "orderId": "219930212",
  "transactionInfo": {
    "amount": 1200,
    "timeStamp": "2014-06-24T15:34:25+00:00",
    "transactionText": "Refrence text",
    "status": "Cancelled",
    "transactionId": "100025255"
  },
  "transactionSummary": {
    "capturedAmount": 0,
    "remainingAmountToCapture": 0,
    "refundedAmount": 0,
    "remainingAmountToRefund": 0
  }
}
```

Image 10: Response Body of a sample API

API definition enables API developers to export API definitions in Swagger and WADL formats from the developer portal. API developers can use these files to generate client-side code by using tools such as Swagger Codegen. They can also import these files into API client tools such as Postman and start calling the API in very little time.

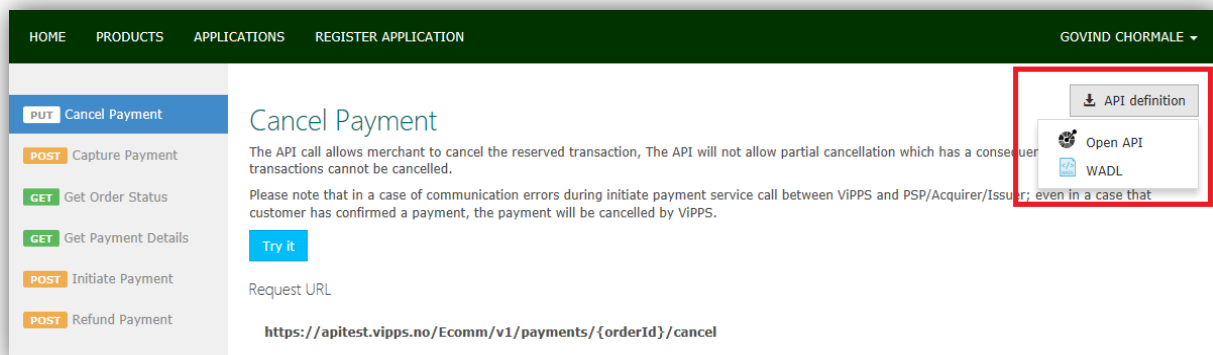


Image 11: API definition to download

3.4 Register an Application

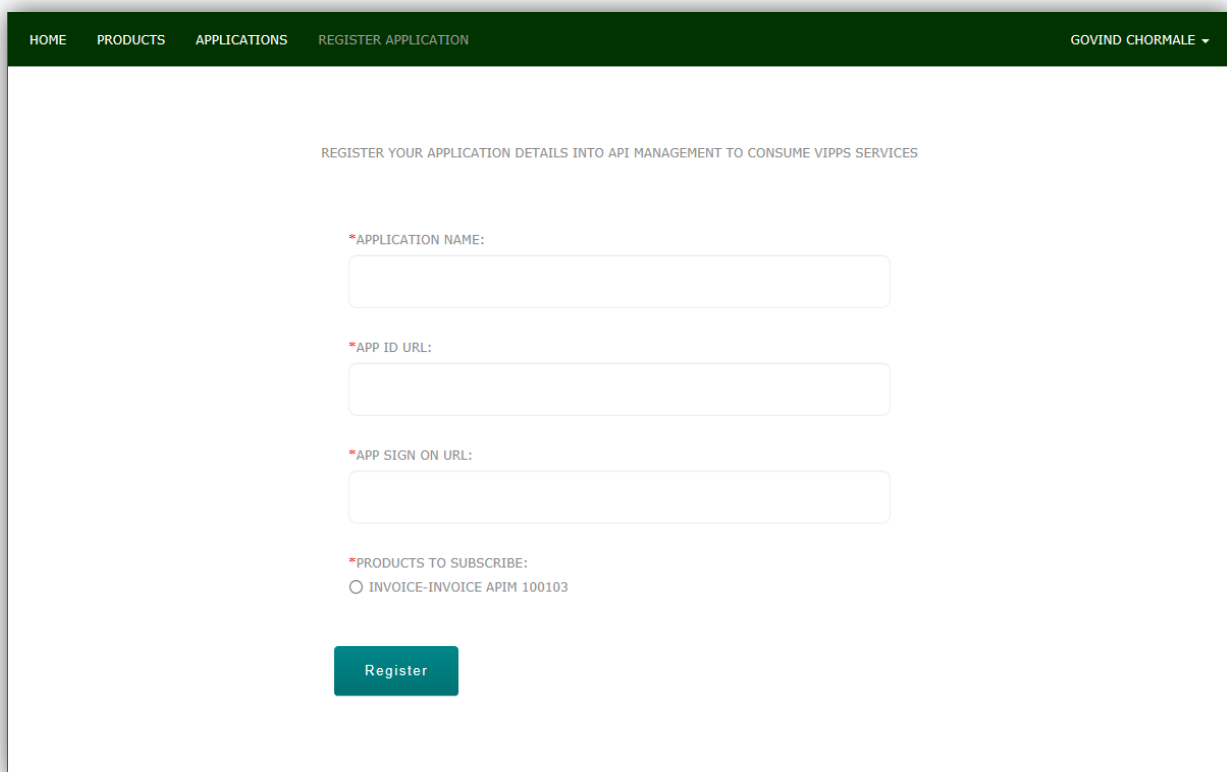
Any application that wants to consume Vipps APIs must be registered first. This registration process involves giving details about merchant application, as described below. Once an application is registered, a unique client Id) and a key client secret will be generated. It is

important to note down the client id and client secret of the application as these parameters are used to get the API access token.

Note: The client secret has an **expiry time of 1 year**. User has to **re-generate** the secret again if the secret is forgotten or expired. Follow [section 3.5.1: Regenerate Client Secret](#) for How to re-generate client secret if it is expired.

The following data should be registered for the application:

- **APPLICATION NAME** – Enter a suitable name for your application to help you identify it.
- **APP ID URL** – This should be a unique URL for the application. For example <https://www.example.com>
- **APP SIGN ON URL** – This should be a unique URL for the application. For example <https://www.example.com/logon>



The screenshot shows the 'REGISTER APPLICATION' page. The header is dark green with navigation links: HOME, PRODUCTS, APPLICATIONS, REGISTER APPLICATION, and a user profile 'GOVIND CHORMALE'. The main content area is white and contains the following form fields:

- *APPLICATION NAME:** A text input field.
- *APP ID URL:** A text input field.
- *APP SIGN ON URL:** A text input field.
- *PRODUCTS TO SUBSCRIBE:** A section with a radio button and the text 'INVOICE-INVOICE APIM 100103'.
- Register** button: A green button at the bottom of the form.

Image 12: Application registration page

On successful registration a client id and secret will be displayed on the page as shown in the picture. Please note them as these are important parameters need to use the APIs.

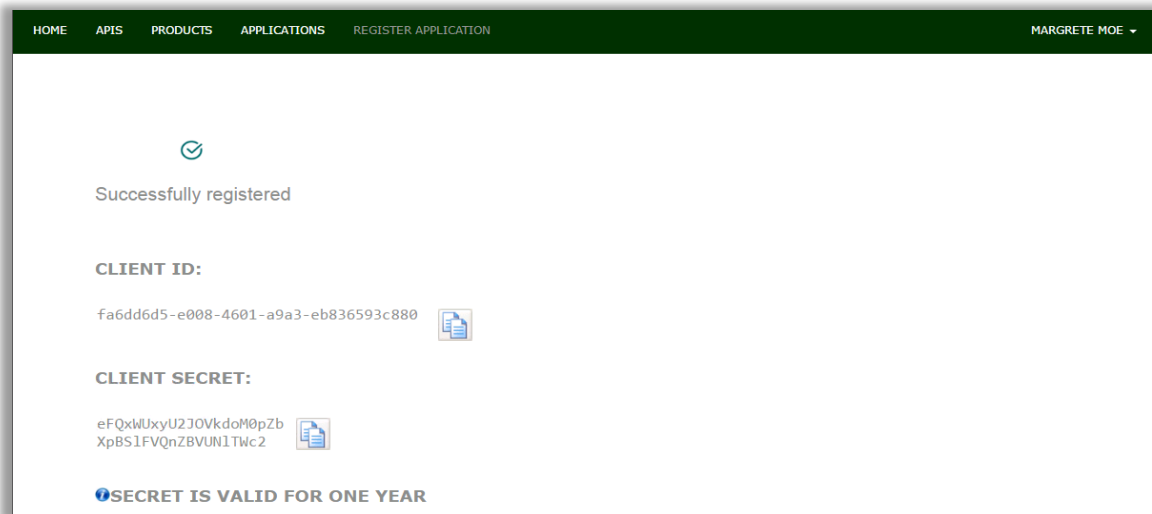


Image 13: Successfully registered

3.5 View registered applications

This is an overview of all the registered applications. Application name and the corresponding products subscribed will be displayed for all the applications registered.

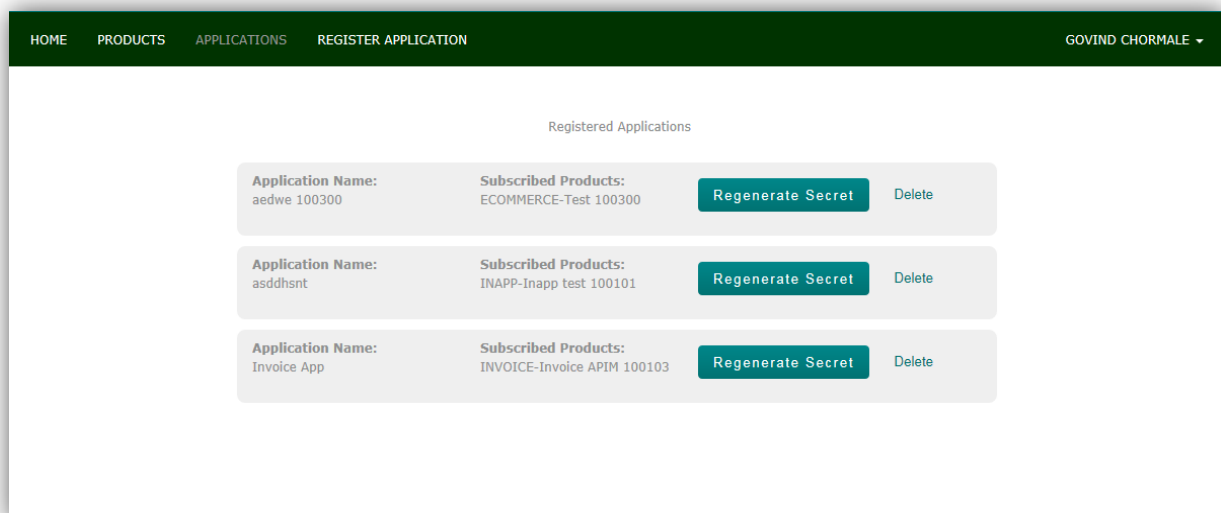


Image 14: View Registered Applications

1. Application secret can be re-generated by clicking the **Regenerate Secret** button.
2. A registered application can be deleted by clicking the **Delete** link.

3.5.1 Regenerate Client secret

User can regenerate client secret if he forgets the secret or if the client secret has been expired, user can regenerate it to use it again. The expiry time for client secret is one year.

1. Click the **Regenerate Secret** button against the application for which secret needs to be regenerated.

2. Regenerated secret along with client id will be displayed on the page as shown in the picture.

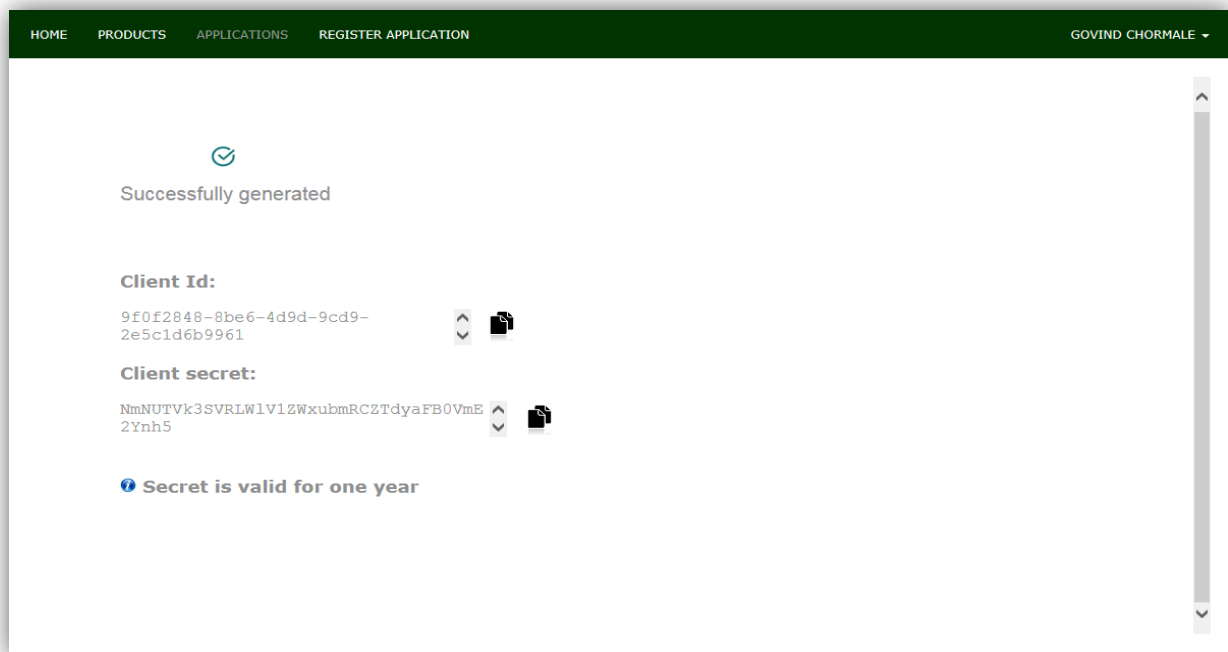


Image 15: Regenerate Secret

3.5.2 Remove a Registered Application

If the merchant does not want to maintain an application, or if he does not want to use any API for a particular application, he can remove a registered application from portal.

1. Click on the Applications tab in navigation bar to navigate to the registered applications page.
2. Click on **Delete** link against the application that needs to be removed.
3. An alert message will be displayed as 'Are you sure you want to delete this item?'
4. Click on OK to remove the application.

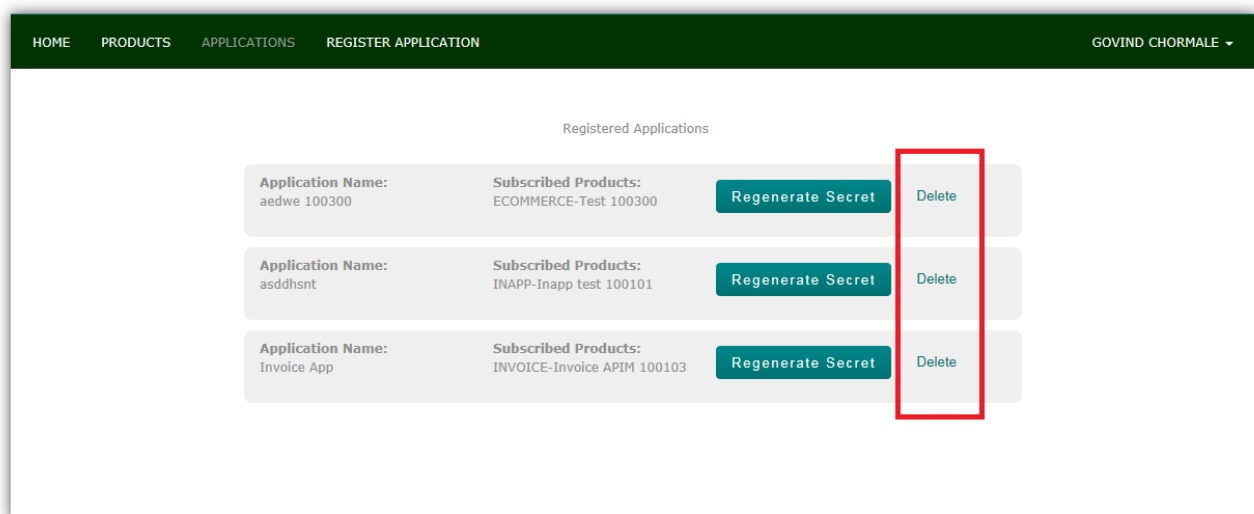


Image 16: Remove a registered Application

3.6 How to Test the APIs

All APIs required to be tested before use. To test an API we need access token in authorization header of the request. Merchant should register at least one application to get the client id and secret that is required for getting the access token. Using Postman or from developer portal, this access token can be acquired.

3.6.1 Get an Access Token

Access token is available in a REST endpoint url. So any REST client tool or application can easily connect to token endpoint to get the access token. Following are some way described how to get an access token

Using developer portal:

1. Click on Product tab in navigation bar to see all the Products.
2. Click on Access Token Product to navigate to access token product page.
3. Click on "TRY IT" button.
4. Enter the client Id, client secret and APIM Subscription key (Subscription key will be automatically filled) in the header and click **Send** button.
5. As a response access token will be received.

Image 14 and 15 shows the request-response body to get access token from developer portal.

HOME PRODUCTS APPLICATIONS REGISTER APPLICATION GOVIND CHORMALE

POST Get Access Token

Access Token

Get Access Token

Use this endpoint to get the access token that needs to be passed in every Vipp's api call. The access token is used for authorizing the request. The maximum lifetime of this token is 24 hours.

Query parameters

+ Add parameter

Headers

client_id Value

client_secret Value

Ocp-Apim-Subscription-Key

+ Add header

Image 17: Access token request parameters

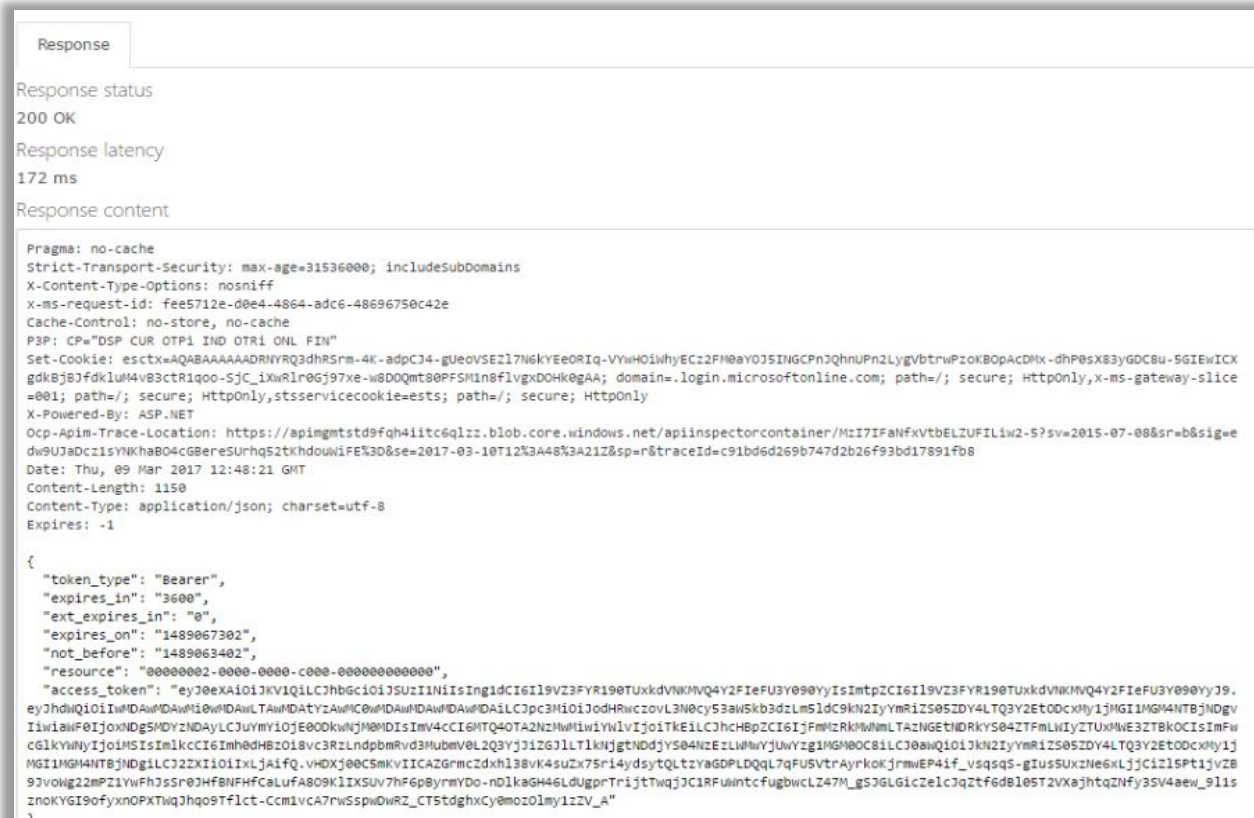


Image 18: Access Token Response

Using Application Code:

The merchants can get an access token using a custom application in any language. Given below are the examples of custom application written in java and C# to get an access token.

Sample code in java:

```
// This sample uses the Apache HTTP client from HTTP Components (http://hc.apache.org/httpcomponents-client-ga/)
import java.net.URI;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.utils.URIBuilder;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class JavaSample
{
    public static void main(String[] args)
    {
        HttpClient httpclient = HttpClients.createDefault();

        try
        {
            URIBuilder builder = new URIBuilder("https://portal.apivipps.no/accessToken/get");
            URI uri = builder.build();
            HttpPost request = new HttpPost(uri);
            request.setHeader("Ocp-Apim-Subscription-Key", "{subscription key}");
            request.setHeader("client_id", "{client_id}");
            request.setHeader("client_secret", "{client_secret}");

            // Request body
            StringEntity reqEntity = new StringEntity("{body}");
```



```

        request.setEntity(reqEntity);

        HttpResponse response = httpClient.execute(request);
        HttpEntity entity = response.getEntity();

        if (entity != null)
        {
            System.out.println(EntityUtils.toString(entity));
        }
    }
    catch (Exception e)
    {
        System.out.println(e.getMessage());
    }
}

```

Sample code in C#:

```

using System;
using System.Net.Http.Headers;
using System.Text;
using System.Net.Http;
using System.Web;

namespace CSHttpClientSample
{
    static class Program
    {
        static void Main()
        {
            MakeRequest();
            Console.WriteLine("Hit ENTER to exit...");
            Console.ReadLine();
        }

        static async void MakeRequest()
        {
            var client = new HttpClient();
            var queryString = HttpUtility.ParseQueryString(string.Empty);

            // Request headers
            client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", "{subscription key}");
            client.DefaultRequestHeaders.Add("client_id", "{client_id}");
            client.DefaultRequestHeaders.Add("client_secret", "{client_secret}");

            var uri = " https://portal.apivipps.no/accessToken/get?" + queryString;

            // Request body
            byte[] byteData = Encoding.UTF8.GetBytes("{body}");

            using (var content = new ByteArrayContent(byteData))
            {
                content.Headers.ContentType = new MediaTypeHeaderValue("< your content type, i.e. application/json
>");

                var response = await client.PostAsync(uri, content);
                var responseContent = await response.Content.ReadAsStringAsync();
                Console.WriteLine("Access Token response: {0}", responseContent);
            }
        }
    }
}

```

Note: The access token lifetime is **1 day (24 hours)**. It can be reused till 24 hours rather than requesting it every time before an API call.

Using Postman:

1. Open the postman from Apps link in Google chrome.

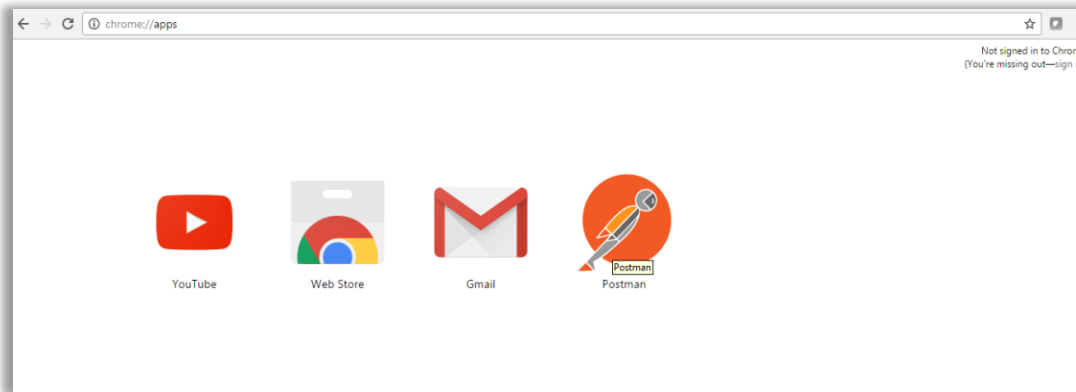


Image 19: Apps Link

2. Enter the token endpoint url (i.e. <https://apitest.vipps.no/accessToken/get>) in postman address bar.
3. Add client_id, client_secret and Ocp-Apim-Subscription-Key in request body as key-value pair. The client_id, client_secret value should be the one received while registering an application. Subscription Key(Ocp-Apim-Subscription-Key) is available in **user profile** page.
4. Change the http verb to '**POST**' and hit Send button. Access token will be received in response as shown below.

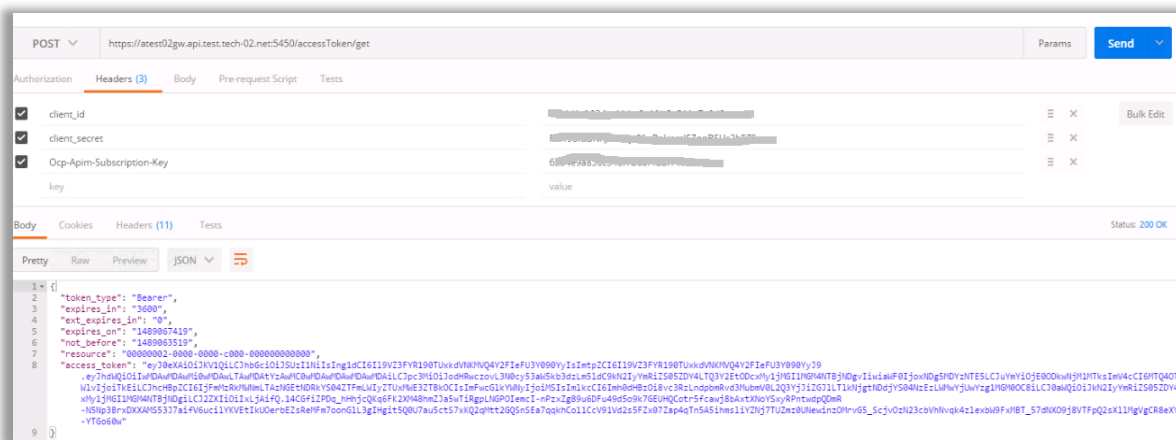


Image 20: Generating access token

After getting access token by any of the above methods please continue with the below steps to test an API.

1. Click on the Product tab in navigation bar to navigate to a Product page and click on "TEST THE API(S)" button to see all the APIs of each and every product.
2. Select the API to be tested and click on "**Try It**" button to test it.
3. Enter the access token in authorization header, subscription key which merchants will get in their profile page for that product and other request parameters as per the API documentation.
4. Finally click on "**Send**" button.

Initiate Payment

initiatePayment

Query parameters

Add parameter

Headers

Content-Type

application/json

Remove header

Ocp-Apim-Subscription-Key

1fe53c389b0e401696cbC

Remove header

Authorization

Bearer eyJ0eXAiOiJKV1Qi

Remove header

X-Request-Id

asis003

Remove header

X-TimeStamp

2016-06-09T13:04:12.1'

Remove header

X-Source-Address

0.0.0.0

Remove header

X-App-Id

d745561a-2059-4941-9e

Remove header

Content-Type

application/json

Remove header

Add header

Image 21: Entering Details

Request URL

https://sit02gw.api.test.tech-02.net/Ecomm/v1/payments

HTTP request

POST https://sit02gw.api.test.tech-02.net/Ecomm/v1/payments HTTP/1.1

Content-type: application/json

Host: sit02gw.api.test.tech-02.net

Ocp-Apim-Subscription-Key:

Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIngldCI6IjE5VzV3FYR190TUXkdVNMkVQ4Y2ZlEiFzU3Y090YyJ9.eyJhdwkiOiWMDAwMDAwMTAwMDAwLTAwMDAtYzAwMC0wMDAwMDAwMDAwMDAwMDA1LCJpc3MiOiJodHRwczovL3N0cy53aW5kb3dzLzI1dC9kMzBkNDE3Zi04MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg1NDUzMTESImV4CCEMTQ4ODU0OTI5MSw1YWIvIjoiTKE1LCJhcHBPZCI6ImQ3NDU1NjFhLTlWNTktNDk0MS05YTIwLWJyOGMwMzI3MmU0ZSIsImFwGClkYmlyIjo1MSIsImklxCI6Imh0dH8z018vc3RzLndpbmRvd3MubmV0L2QzMGQ0MTdmLTgwOTATNDM2NS1hZmYwLTA0MjBjBjMzYzODh0Ni81LjE0MDkxLTQzNjU0Y2ZlYyYiIiwiaWF0IjoxNDg0NTQ1MzE0LjUyYyIjE0ODg

Image 22: Request URL & HTTP request

5. Merchants will be getting a response as per the API definition document.

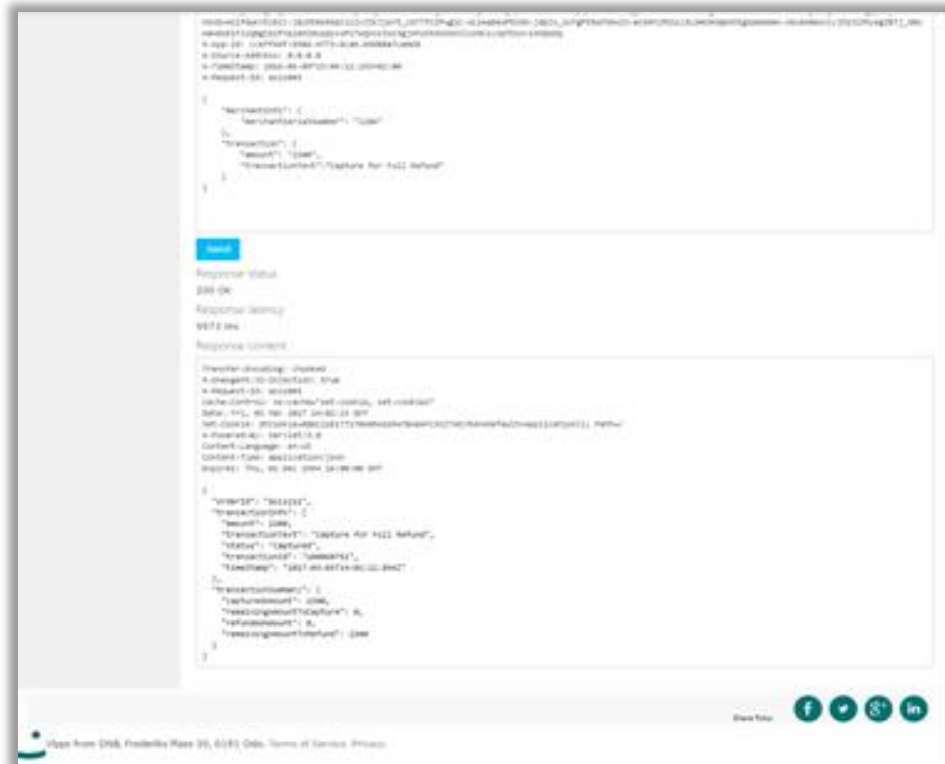


Image 23: Response body

3.7 Reports & Analytics

The Analytics Reports section provides a detailed metrics and analytics report on various information like usage and health of APIs and operations, activities performed by developer, products etc. It has the following four tabs:

- **At a glance:** Provides overall usage and health metrics
- **Usage:** Provides an in-depth look at API calls and bandwidth
- **Health:** Provides data for usage over time for cache, response time and status codes
- **Activity:** Provides statistics for success rates for API calls and average response time

4. How to call Vipps APIs from application

As described in earlier sections, any application must pass the APIM subscription key and access token in request header to call a Vipps API. The access token has an expiry time of max. 1 day (24 hours).

The following pseudo code samples would help a developer how to get an access token and use it to call an API from a custom application. It also shows the way of regenerating the access token and uses it for API calls once previous token is expired.

4.1 Define an Application Cache

To reuse access token we need to temporarily store it somewhere, application can easily get it. It is better to store the token in application cache like InMemory or Redis Cache service for faster access.

Cache pseudo code:

```
public class Cache
{
    var cacheClient = new CacheClient("cache connection details");

    public string GetAccessToken(string key)
    {
        return cacheClient.GetValue(key) as string;
    }

    public void SaveAccessToken(string key, string token)
    {
        cacheClient.SetValue(key, token);
    }
}
```

2.2 Get Access Token

Following code sample shows how to get an access token and save it in cache to reuse until it expires. To get the access token we need to call the "<https://apitest.vipps.no/accessToken/get>" endpoint, given in the developer portal. The "isNew" argument decides whether a new token will be generated or it will be fetched from cache. Access token endpoint url requires following parameters in the request.

- SubscriptionKey – This will be found in developer portal
- ClientId and ClientSecret – You will get these values when you register an application as stated in [section 3.3 : Register an Application](#)

GetAccessToken pseudo code:

```
var token = string.Empty();
var client = new HttpClient();
var queryString = HttpUtility.ParseQueryString(string.Empty);
var cache = new Cache();

public async string GetAccessToken(bool isNew)
{
    If(isNew)
    {
        // Request headers
        client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", "{subscription key}");
        client.DefaultRequestHeaders.Add("client_id", "{client_id}");
        client.DefaultRequestHeaders.Add("client_secret", "{client_secret}");

        var uri = " https://portal.apivipps.no/accessToken/get?" + queryString;

        // Request body if any
        byte[] byteData = Encoding.UTF8.GetBytes("{body}");

        using (var content = new ByteArrayContent(byteData))
        {
            content.Headers.ContentType = new MediaTypeHeaderValue("application/json");
            var response = await client.PostAsync(uri, content);
            var responseContent = await response.Content.ReadAsStringAsync();

            //Get access token from responseContent
            token = responseContent.{AccessToken}

            // Save token in cache [Optional]
            cache. SaveAccessToken("AccessToken", token);
        }
    }
}
```

```

        return token;
    }
    Else
    {
        return cache. GetAccessToken("AccessToken");
    }
}

```

2.3 Call an API

Once you get an access token, now you can call an API. Before calling an API from an application you can test it developer portal as stated in [section 3.4: How to Test the APIs](#). Follow this code sample below for how to call a Vipps API.

API calls pseudo code:

```

var client = new HttpClient();
var queryString = HttpUtility.ParseQueryString(string.Empty);

public void SetUpCommonRequestHeaders()
{
    // Request headers
    client.DefaultRequestHeaders.Add("Content-Type", "application/json");
    client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", "{product subscription key}");
    client.DefaultRequestHeaders.Add("X-App-Id", "{client_id}");
    client.DefaultRequestHeaders.Add ("X-Request-Id", "asis003");
    client.DefaultRequestHeaders.Add ("X-TimeStamp", "2016-06-09T13:04:12.193+02:00");
    client.DefaultRequestHeaders.Add ("X-Source-Address", "0.0.0.0");

    return client;
}

Public async IEnumerable<Payment> GetPayments()
{
    //*****First API call*****//

    var accessToken = GetAccessToken(true); // For 1st time pass true to create a new access token
    var response = await GetHttpResponse (accessToken);

    if(response.StatusCode == SatusCode.Ok) // Status code 200
    {
        var payments = await response.Content.ReadAsAsync<IEnumerable<Payment>>();
        return payments;
    }
    else
    {
        return null;
    }

    //*****Next API call*****//

    var accessToken = GetAccessToken(false); // Get access token from cache; don't create new
    var response = await GetHttpResponse (accessToken);

    if(response.StatusCode == SatusCode.Ok) // Status code 200
    {
        var payments = await response.Content.ReadAsAsync<IEnumerable<Payment>>();
        return payments;
    }
    else
    {
        return null;
    }
}

```

*******API call when token is EXPIRED*******

```

var accessToken = GetAccessToken(false); // Get access token from cache; don't create new
var response = await GetHttpResponse (accessToken);

if(response.StatusCode == SatusCode.Ok) // Status code 200
{
    var payments = await response.Content.ReadAsAsync<IEnumerable<Payment>>();
    return payments;
}
else if(response.StatusCode == SatusCode.Unauthorize && response.Message == "Access token is missing
or invalid") // Status code 401 and token is invalid as it may be expired
{
    accessToken = GetAccessToken(true); // create new access token and save in cache again.
    response = await GetPayments (accessToken);
    if(response.StatusCode == SatusCode.Ok) // Status code 200
    {
        var payments = await response.Content.ReadAsAsync<IEnumerable<Payment>>();
        return payments;
    }
    else
    {
        return null;
    }
}
}

public async HttpResponseMessage GetHttpResponse(string token)
{
    // Set common headers
    SetUpCommonRequestHeaders();

    // Set Authorization header with access token
    client.DefaultRequestHeaders.Add("Authorization","Bearer " + accessToken);

    var apiUrl = new Uri("https://apitest.vipps.no/Ecomm/v1/payments");

    // Request body if any
    byte[] byteData = Encoding.UTF8.GetBytes("{body}");

    using (var content = new ByteArrayContent(byteData))
    {
        content.Headers.ContentType = new MediaTypeHeaderValue("application/json");
        return await client.PostAsync(apiUrl, content);
    }
}

```