**Vipps eCommerce APIs v2**

Document version: **2.0**

Date produced: **Nov-2017**

Content

[Overview 4](#_Toc506283608)

[1. Use case scenarios 4](#_Toc506283609)

[1.1 Regular eCommerce Payments 4](#_Toc506283610)

[1.2 Express Checkout Payments 4](#_Toc506283611)

[2. API Calls flow 5](#_Toc506283612)

[3. Authentication 6](#_Toc506283613)

[3.1 Overview 6](#_Toc506283614)

[3.2 Access Token 6](#_Toc506283615)

[4. Idempotency 8](#_Toc506283616)

[5. eCommerce Payment Flows 9](#_Toc506283617)

[5.1 Initiate 9](#_Toc506283618)

[5.2 Reserve 9](#_Toc506283619)

[5.3 Cancel 9](#_Toc506283620)

[5.4 Capture 9](#_Toc506283621)

[5.5 Direct capture 10](#_Toc506283622)

[5.6 Refund 10](#_Toc506283623)

[5.7 Get Order Status 10](#_Toc506283624)

[5.8 Get Payment Details 10](#_Toc506283625)

[6. Additional payment flow for express checkout 10](#_Toc506283626)

[6.1 Get shipping cost & method 10](#_Toc506283627)

[6.2 Transaction updates with user details 10](#_Toc506283628)

[6.3 Remove user consent 11](#_Toc506283629)

[7. Exception handling 11](#_Toc506283630)

[7.1 Introduction 11](#_Toc506283631)

[7.2 Exception scenarios 11](#_Toc506283632)

[8. Response codes 12](#_Toc506283633)

[8.1 Success Codes 12](#_Toc506283634)

[8.2 Error Codes 12](#_Toc506283635)

[8.3 Error Representation 13](#_Toc506283636)

[8.4 Error codes 13](#_Toc506283637)

[9. Front-end Integration 13](#_Toc506283638)

[9.1 Appswitch between Mobile or Desktop Browser and Vipps App 14](#_Toc506283639)

[9.2 Appswitch between Merchant’s Mobile App and Vipps App 14](#_Toc506283640)

[9.3 List of error codes for deeplinking 20](#_Toc506283641)

[10. API definitions 20](#_Toc506283642)

[10.1 Request Headers 20](#_Toc506283643)

[10.2 Payment sequence 21](#_Toc506283644)

[10.3 Initiate Payment 23](#_Toc506283645)

[10.4 Cancel Payment 29](#_Toc506283646)

[10.5 Capture Payment 32](#_Toc506283647)

[10.6 Refund Payment 34](#_Toc506283648)

[10.7 Get Payment Details 36](#_Toc506283649)

[10.8 Get Order Status 39](#_Toc506283650)

[10.9 Fetch Shipping Cost & Method (Hosted by Merchant for express checkout) 40](#_Toc506283651)

[10.10 Callback : Transaction Update 41](#_Toc506283652)

[10.11 Remove User Consent (for express checkout) 43](#_Toc506283653)

[11. Appendix 44](#_Toc506283654)

# Overview

Vipps eCommerce API gives merchant a great control over Vipps payment lifecycle. It gives possibility to initiate payment from mobile app and webshop. It also enables merchant to affect payment flow by utilizing functions like payment reservation, capture, cancellation and refunding.

# Use case scenarios

In order to ease integration with Vipps and get better understanding of API functionality and usage some typical scenarios is presented.

## Regular eCommerce Payments

Merchants can utilize vipps payment services for ecommerce payments at their online websites. Vipps APIs can be integrated at merchant’s checkout page where vipps can be represented as one of the payment methods for online purchases.

Payment initiation can happen on desktop browser, mobile browser as well as in merchant’s mobile app. This latest version of API document explains how we have simplified development on merchant side. Vipps takes responsibility of identifying whether user is on mobile browser or desktop browser and whether user has vipps app in device or not. This is going to be possible by introduction of “Vipps landing page”. You can read more description about initiation of Vipps ecommerce payment [here.](#_Initiate_Payment)

Introduction of landing page will simplify merchant’s implementation of ecommerce APIs in general.

Vipps ecommerce APIs support both direct sale as well as reservation-capture logic in API implementation. Merchant needs to get this configured during enrollment. One vipps ecommerce product supports one of the two approaches (reservation-capture or direct sale).

Following are different API services that merchant can utilize as part of ecommerce payments:

Initiate Payment – Used for initiating ecommerce payment

Cancel Payment – Optional. Used for cancel payment reservation

Capture Payment – Used for capture reserved payment

Refund Payment – Optional. Used for refund payments

Get Payment Details – Optional. Used to retrieve transaction history

Get Order Status – Optional. Server side check that payment is confirmed

## Express Checkout Payments

In addition to regular ecommerce APIs, Vipps also supports optional express checkout payment service. Merchant needs to be onboarded for this additional service as it involves sharing personal information of user (including address), shipping details, etc.

Vipps follows GDPR compliance for making express checkouts possible for merchants. Vipps takes away the lengthy checkout process to simple steps in Vipps app which gives merchants higher conversion rates of online purchases.

Vipps can share personal information of Vipps users with merchant by taking user’s consent as per GDPR law during checkout process. Also, it will be possible to populate different delivery options to Vipps users during checkout within Vipps app.

Upon user’s final confirmation of payment, Vipps will share payment information, address and shipping method information and Vipps user’s personal information (optional) once payment is processed.

In addition to above mentioned ecommerce services, merchants can also implement the below services in order to use express checkout feature.

Get shipping cost & method – Used for fetching shipping cost & method combination based on user selected address

Transaction update with user details – separate callback to merchant for receiving post payment information

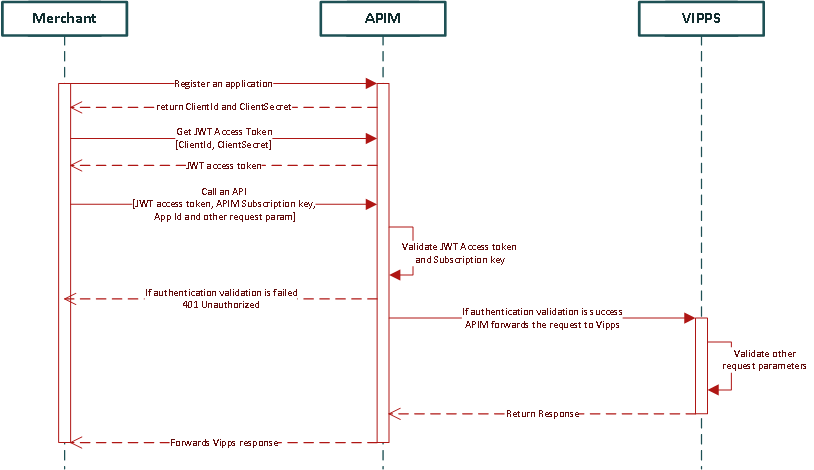
Remove user consent – Used to inform merchant when vipps user removes consent to share his details.

# API Calls flow

This section will explain how merchants can start using Vipps APIs and get access to API credentials.

During merchant’s onboarding process in Vipps, the merchant receives a username and a password to login into the Merchant Developer Portal (manual to use merchant developer portal can be found [here](#_Appendix)). Once logged in to the developer portal, the merchant needs to register an application to generate a JWT access token and use the token in authorization header of the request.

The diagram below shows the integration flow between merchant and Vipps server.



All communication with the Vipps ecommerce API has to be authenticated via JWT access token. To get this access token and use it in API calls merchant should follow the steps below:

* Merchant logs into the Developer portal and registers an application that will consume Vipps APIs. On successful registration, it will receive application credentials (ClientId and ClientSecret).
* Merchant application uses the clientid and clientsecret to get a JWT access token from APIM. JWT access token is a base 64 encoded string value that needs to be used as a bearer token in the request header.
* Merchant application will have to use this JWT access token, APIM subscription key and App Id(clientid) along with other request parameters while calling a Vipps API.
* APIM validates the JWT access token and subscription key. If token is invalid it sends 401 unauthorized while if it is valid, request is forwarded to Vipps.

Vipps process the request and produce corresponding response which is sent back to merchant application via APIM.

# Authentication

## Overview

Every API call is authenticated and authorized based on the application access token (JWT Bearer token) and APIM subscription key (Ocp-Apim-Subscription-Key). In addition, APP Id need to be provided in request header in order to authenticate the application the merchant has registered. Following headers are required to be there in every API request to successfully authenticate every API call.

|  |  |  |
| --- | --- | --- |
| Header Name | Header Value | Description |
| Authorization | “Bearer <jwt access token>” | type: Authorization token  Value: Access token is obtained by registering merchant backend application in Merchant Developer Portal. |
| Ocp-Apim-Subscription-Key | Base 64 encoded string | Subscription key for eCommerce product. This can be found in User Profile page on Merchant developer portal after merchant account is created |

## Access Token

### Overview

Access token API endpoint helps to get the JWT Bearer token that needs to be passed in every API request in the authorization header. Merchant application use the <**ClientId**> and <**ClientSecret**> to get a JWT access token. JWT access token is a base 64 encoded string value that must be aquire first before making any Vipps api calls.

### URL

https://<<hostname>>/accessToken/get

### Method

POST

### Request Headers

|  |
| --- |
| “client\_id”:<ClientID>  “client\_secret”:<ClientSecret>  “Ocp-Apim-Subscription-Key”:<Ocp-Apim-Subscription-Key> |

### Description

|  |  |  |  |
| --- | --- | --- | --- |
| Header Name | Header Value | Optional | Description |
| client\_id | A GUID value | No | Client ID received when merchant registered the application |
| Client\_secret | Base 64 string | No | Client Secret received when merchant registered the application |
| Ocp-Apim-Subscription-Key | Base 64 encoded string | No | Subscription key for Access Token product which is subscribed by default. This can be found in User Profile page on Merchant developer portal |

### Success Response

|  |  |
| --- | --- |
| Http Status Code | Content |
| 200 | OK |

### Success Response Body

|  |
| --- |
| {  "token\_type": "Bearer",  "expires\_in": "86398",  "ext\_expires\_in": "0",  "expires\_on": "1495271273",  "not\_before": "1495184574",  "resource": "00000002-0000-0000-c000-000000000000",  "access\_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6InowMzl6ZHNGdWl6cEJmQlZLMVRuMjVRSFlPMCIsImtpZCI6InowMzl6ZHNGdWl6cEJmQlZLMVRuMjVRSFlPMCJ9..QaopYuvN8jsh82uepLcF-uLqEhdFsRNl6\_KrjAva537HHMa2x6w2pL2v96k40QBjD8A\_GGHZ-E2VC3QSY-WsPdHUI5Kb4zEQzJ4-\_CnMRo3bXavz3Sdo2-1amFKsOY8AFODpqJR0MYqPK\_Kr6sSIWL3M\_L3wu0rG976HIXllsRLvWBSwDeMgBAUvwWrXCmnVfznOleSxsPbAxZshn3xjuYeJWEAR7ZpJQhjuGpiu0rP7lERTxX\_rCnW1cr3m7RfEl6z8e5VQva9AOt4OG5NuIrLJqmhb3KHBa3GusK6KLf-pVjF6fnS5r0ZQ5foP-VqOCiK9CUUATjHEOflgy1ubvA"  } |

### Description

|  |  |  |
| --- | --- | --- |
| Id | Type | Description |
| token\_type | String | It’s a bearer type token. When used the word ‘Bearer’ must be added before the token value |
| expires\_in | Integer | Token expiry duration in seconds |
| ext\_expires\_in | Integer | Any extra expiry time. This is zero only |
| expires\_on | Integer | Token expiry time in epoch time format |
| not\_before | Integer | Token creation time in epoch time format |
| resource | GUID String | A common resource object that comes by default. Not used in token validation |
| access\_token | Base 64 String | The actual access token that needs to be used in request header |

### Error Response

|  |  |  |
| --- | --- | --- |
| Http Status Code | Content | Description |
| 400 | Bad Request | If ClientId is invalid |
| 401 | Unauthorized | If ClientSecret is invalid |
| 5xx | Internal server error | Internal server error |

### Error Response Body

#### 400 Bad Request Error

|  |
| --- |
| {  "error": "unauthorized\_client",  "error\_description": "AADSTS70001: Application with identifier 'e9b6c99d-2442-4a5d-84a2-c53a807fe0c4' was not found in the directory testapivipps.no\r\nTrace ID: 3bc2b2a0-d9bb-4c2e-8367-5633866f1300\r\nCorrelation ID: bb2f4093-70af-446a-a26d-ed8becca1a1a\r\nTimestamp: 2017-05-19 09:21:28Z",  "error\_codes": [  70001  ],  "timestamp": "2017-05-19 09:21:28Z",  "trace\_id": "3bc2b2a0-d9bb-4c2e-8367-5633866f1300",  "correlation\_id": "bb2f4093-70af-446a-a26d-ed8becca1a1a"  } |

#### 401 Unauthorized Error

|  |
| --- |
| {  "error": "invalid\_client",  "error\_description": "AADSTS70002: Error validating credentials. AADSTS50012: Invalid client secret is provided.\r\nTrace ID: 7ca46a74-8ef0-4a01-8bb1-c5a277f00a00\r\nCorrelation ID: 778bf4a1-5d91-4f74-bb3f-7f4541f1ccd2\r\nTimestamp: 2017-05-19 09:23:52Z",  "error\_codes": [  70002,  50012  ],  "timestamp": "2017-05-19 09:23:52Z",  "trace\_id": "7ca46a74-8ef0-4a01-8bb1-c5a277f00a00",  "correlation\_id": "778bf4a1-5d91-4f74-bb3f-7f4541f1ccd2"  } |

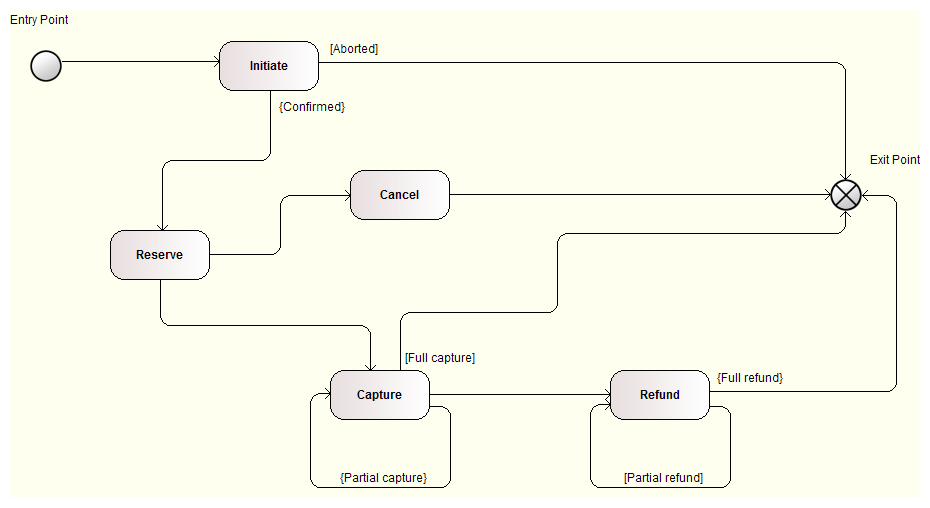
# Idempotency

All API requests in Vipps eCommerce can be retried without any side effects by providing idempotent key in a header of the request. For example, in case the request fails because of network error it can safely be retried with the same idempotent key. Idempotent key is generated by merchant.

-H "X-Request-Id: slvnwdcweofjwefweklfwelf"

# eCommerce Payment Flows

Payment flow in Vipps eCommerce is represented by following state diagram:



## Initiate

First call in payment flow initiates payment request that is subject of customer (end user) confirmation. Payment has status *Initiated* and customer is notified about payment request in mobile app. If customer doesn’t confirm, the payment request cancels and payment flow aborts. Initiate call will have parameter *paymentType* which will identify regular ecommerce payment and express checkout payment flow.

## Reserve

When the customer successfully authorizes the payment request by using Vipps app, the payment status changes to *Reserved*, and the respective amount will be reserved for future capturing at the PSP.

## Cancel

Reservations can be cancelled and payment flow aborted under certain circumstances:

* When user cancels the initiated payment then the payment status shown will be cancelled
* When Merchant call cancellation of the reservation. Please note that partially captured reservations can’t be cancelled.
* When there is a timeout of the payment confirmation by several of reasons (no action by the customer, notification to user is delayed, etc.)
* When reservation fails caused by system or communication error.

## Capture

When merchant shipped the goods then they can call capture API on the reserved transaction. The API allows to do a full amount capture or partial amount capture.

## Direct capture

Direct capture is not depicted on diagram above but, in essence, combines two steps (reserve and capture) in one. This is a configuration in Vipps backend when Initiate payment request is sent.

## Refund

Merchant can initiate a refund of the captured amount. The refund can be a partial or full.

## Get Order Status

Get Order Status intention is to check whether the user is authenticated the transaction or not. Possible status provided by this service is listed below:

|  |  |
| --- | --- |
| Status | Description |
| INITIATE | Initiate Payment |
| REGISTER | When we call PSP for payment |
| RESERVE | Parent Reserve Payment and for its child Capture\Refund payments |
| SALE | Direct Capture |
| CANCEL | When payment has been cancelled |
| VOID | Registration Cancel |
| AUTOREVERSAL | Transaction timed out at Nets, we will refund the transaction and then it will be autoreversal |
| AUTOCANCEL | When no action from user for notification for X minutes. |
| FAILED | When transaction failed to execute |
| REJECTED | When user reject payment request in Vipps app |

## Get Payment Details

Get Payment Details will provide all the payment transaction details.

# Additional payment flow for express checkout

In addition to above mentioned payment flows, following are the services which merchants should build on their side to support express checkout during online purchases.

## Get shipping cost & method

When express checkout payment is initiated, vipps will call this service from merchant’s backend to fetch shipping cost and shipping method related details. Merchant can send priority of shipping cost and method combination if there are multiple ways of delivery. Merchant can also send default shipping cost & method combination which merchant wants user to see on payment confirmation screen of Vipps. Vipps will support upto 10 shipping cost and method combinations. If user sends more than 10 combinations, vipps will display first 10 always.

## Transaction updates with user details

After express checkout payment is processed, vipps will make a call back to merchant stating payment details, shipping details and user details (optional).

## Remove user consent

When consent to store/process/view details of vipps user is removed by user in vipps app, vipps will make a call to merchant informing the same. Merchant is obliged to delete user information upon receiving this request.

# Exception handling

## Introduction

Every system, especially those that includes complex integrations and/or participation of many users, is prone to unexpected conditions. Below section explains how Vipps handles different exception and error situations in detail.

## Exception scenarios

The most critical action in payment flow is when Initiate Payment service call is invoked. The Flow diagram below shows how to successfully fulfil service call, communication between several contributors and users across several systems has to work flawlessly.



To cope with possible communication problems/errors, several scenarios and guidelines are developed.

### Connection timeout

Defining a socket timeout period is the common measure to protect server resources and is expected. However, the time needed to fulfill a service requests depends on several systems, which impose longer timeout period than usually required. We recommend setting no less than 1 second socket connection timeout and 5 seconds socket read timeout while communicating with Vipps.

A good practice is, if/when the socket read timeout occurs, then call Get Payment Details and check status of last transaction in transaction history prior executing the service call again.

### Callback aborted/interrupted

If the communication is broken during payment process for some reason, and Vipps is not able to execute callback, then callback will be retried after sometime within a callback timeframe (timeframe can be for example 10 minutes).

In other words, if the merchant doesn’t receive any confirmation on payment request call within callback timeframe, merchant should call get payment details service to get the response of payment request.

### PSP connection issues

In a case when Vipps experiences communication problems with PSP, service call will respond with 402 HTTP Error. In such cases, Vipps will automatically retry last transaction request (capture/cancel/refund) against PSP. Merchant should always make a call to Get Payment Details to check if the transaction request is processed before making service call (with same idempotency key) again.

# Response codes

Vipps eCommerce API uses standard HTTP response codes to indicate the success and failure of the request as defined in RFC2616 (<https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>). Response codes with range 2xx indicates success, 4xx indicates an error because (validation error, Reservation of transaction failed etc.), 5xx are the vipps internal errors.

## Success Codes

200 – OK  
202 - Accepted

## Error Codes

400 – Bad request (Missing a required parameter or Bad request formats)  
401 – Unauthorized Vipps

403 – Forbidden  
404 – Resource Not Found

405 – Request method not supported

415 – Unsuppoted media type  
5XX – Something went wrong from Vipps Server side

In the case of error, body of response contains detailed information about the error condition. Error object is represented in JSON format as:

[{  
 "errorCode": "",  
 "errorMessage": ""  
 }]

|  |  |  |
| --- | --- | --- |
| Field | example | description |
| errorCode | 01-100 | error code which uniquely identifies an error scenario |
| errorMessage | "Authentication Failed" | Error message to display |

## Error Representation

|  |  |
| --- | --- |
| Error Groups | Description |
| Authentication | Authentication Failure because of wrong token provided |
| Payment | Failure while doing a payment Authorization, mostly because of PSP errors |
| InvalidRequest | Request contains invalid parameters |
| VippsError | Internal Vipps application error |
| Customer | Error raised because of Vipps user (Example: User not registered with Vipps ....) |
| Merchant | Errors regarding the merchant |

## Error codes

|  |  |  |
| --- | --- | --- |
| Error Group | Error Code | Error Message |
| Payment | 41 | User don’t have a valid card |
| Payment | 42 | Refused by issuer bank |
| Payment | 43 | Refused by issuer bank because of invalid amount |
| Payment | 44 | Refused by issuer because of expired card |
| Payment | 45 | Reservation failed for some unknown reason |
| Payment | 51 | Can't cancel already captured order |
| Payment | 52 | Cancellation failed |
| Payment | 53 | Can’t cancel order which is not reserved yet |
| Payment | 61 | Captured amount exceeds the reserved amount ordered |
| Payment | 62 | Can't capture cancelled order |
| Payment | 63 | Capture failed for some unknown reason, please use Get Payment Details API to know the exact status |
| Payment | 71 | Cant refund more than captured amount |
| Payment | 72 | Cant refund for reserved order, use cancellation API for the same |
| Payment | 73 | Can't refund on cancelled order |
| Payment | 74 | Refund failed during debit from merchant account |
| InvalidRequest | {field\_name will be the error code} | Description about what exactly the field error is |
| VippsError | 91 | Transaction is not allowed |
| VippsError | 92 | Transaction already processed |
| VippsError | 98 | Too many concurrent requests |
| VippsError | 99 | Description about the internal error |
| Customer | 81 | User Not registered with vipps |
| Customer | 82 | User App Version is not supported |
| Merchant | 31 | Merchant is blocked because of {} |
| Merchant | 32 | Receiving limit of merchant has exceeded |
| Merchant | 33 | Number of payment requests has been exceeded (Not used) |
| Merchant | 34 | Unique constraint violation of the order id |
| Merchant | 35 | Requested Order not found |
| Merchant | 36 | Merchant agreement not signed |
| Merchant | 37 | Merchant not available or deactivated or blocked |
| Merchant | 21 | Reference Order ID is not valid |
| Merchant | 22 | Reference Order ID is not in valid state |

# Front-end Integration

Merchants need to implement “Appswitch” integration which is also called as “Deeplinking” to trigger Vipps app for serving ecommerce payment requests. This can happen in two ways:

1. From mobile or desktop browser
2. From mobile application

Below section explains how merchant can implement these integrations in detail.

## Appswitch between Mobile or Desktop Browser and Vipps App

* For mobile/desktop browser to Vipps front end integration, merchant doesn’t need to do any special activity.
* Front end integration will be handled by Vipps using Vipps landing page.
* Merchant needs to ensure passing correct “fallbackURL” in Vipps backend API (explained [here](#_Initiate_Payment)).
* After vipps has completed the operation the “fallbackURL” will be opened in a new tab/new window in the browser. To maintain the session, merchant can pass along a session identifier through “fallbackURL”.

## Appswitch between Merchant’s Mobile App and Vipps App

App to App switch is supported by both Vipps applications on iOS and Android platform. The two subsections below explain how appswitch happens for iOS and Android respectively.

### Appswitch with iOS platform

Following section explains how appswitch will happen for Vipps app on iOS platform.

#### Overview for iOS

* Vipps app on iOS platform requires URL scheme in order to support appswitch.
* Merchant need to pass the URI Scheme of app into “fallbackURL” in Vipps backend API (explained [here](#_Initiate_Payment)).
* Merchant will open the url received from Vipps backend API.
* Once the operation in Vipps is completed, Vipps will open the url mentioned in “fallbackURL”.
* From vipps mobile application appropriate status code will be appended with “fallbackURL”.

#### Switch from Source App to iOS Vipps App

Below is sample code to open iOS Vipps application with deeplinkURL.

|  |
| --- |
| NSString \*url = deeplinkURL; *//Use deeplink url provided in API response*  if ([[UIApplication sharedApplication] canOpenURL:[NSURL URLWithString:url]]) {  [[UIApplication sharedApplication] openURL:[NSURL URLWithString:url]];  }  else {  *// Oops no Vipps app or update to latest Vipps App! Open app store page. Once user installs Vipps, calling app needs to initiate deeplinking again in order to get the callback*  [[UIApplication sharedApplication] openURL:[NSURL URLWithString: @"https://itunes.apple.com/no/app/Vipps-by-dnb/id984380185"]];  } |

#### Redirect Back to Source App from iOS Vipps App

Once the operation in Vipps is completed, vipps mobile application will open the frontend url. For app to app integration, merchant app needs to be registered for a url scheme and pass the url scheme in “fallbackURL” in Vipps backend API (explained [here](#_Initiate_Payment)). Vipps mobile application will use below code to launch merchant application.

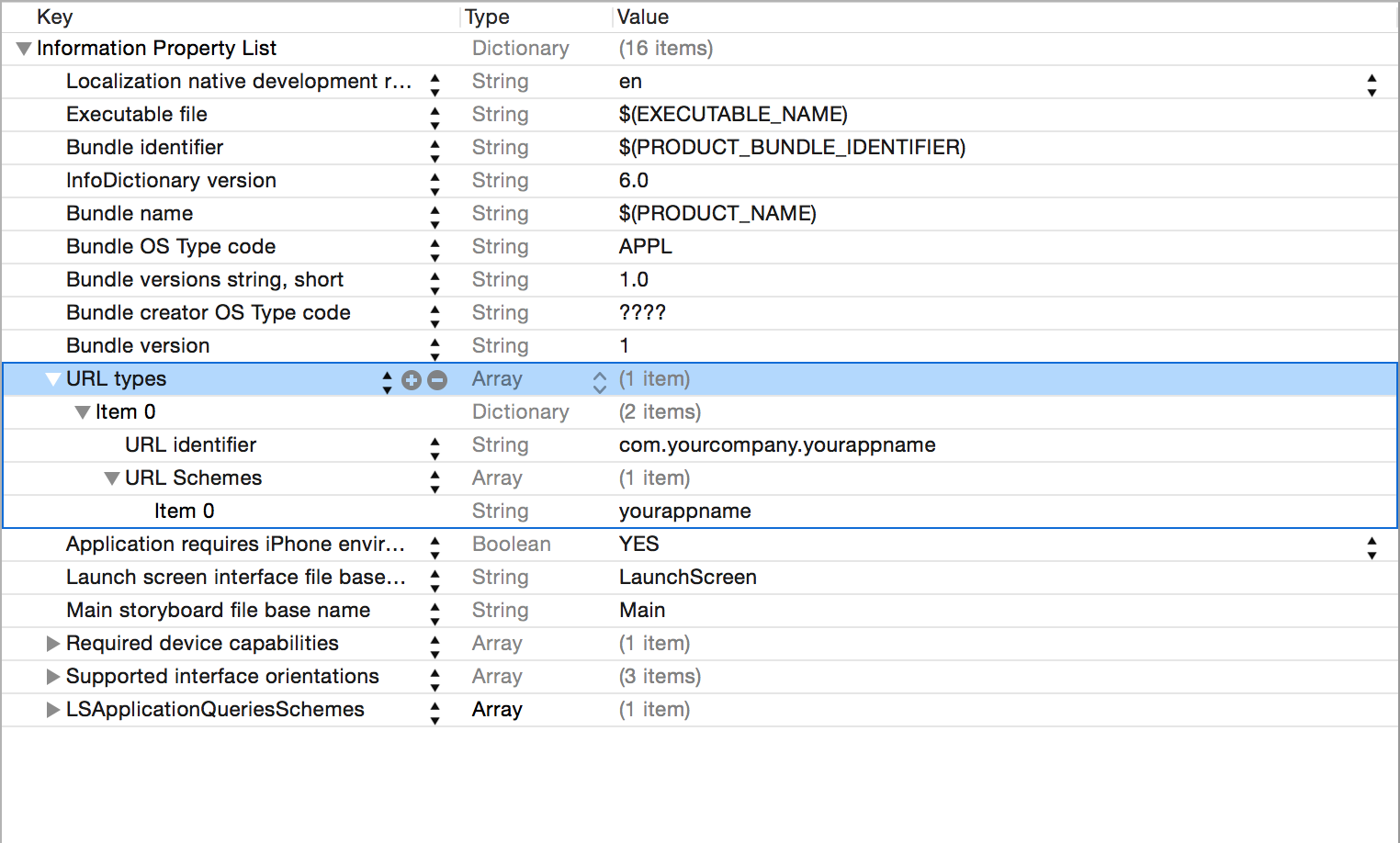
|  |
| --- |
| NSString \*fallbackURL = self.fallbackURL; *//fallback url will be the url which has been provided in Vipps API.*  NSURLComponents \*urlComponents = [NSURLComponents componentsWithString:fallbackURL];  NSMutableArray <NSURLQueryItem \*>\*queryItems = [[NSMutableArray alloc]initWithArray:urlComponents.queryItems];  NSURLQueryItem \*statusQueryItem = [NSURLQueryItem queryItemWithName:@"status" value:@"301"];  *//Add the queryitem in the “queryItems” array.*  [queryItems addObject:statusQueryItem];    NSURL \* fallbackURL = urlComponents.URL;*//after adding the new queryItems we will get the new fallbackURL*  *// navigating back to source application*  UIApplication \*application = [UIApplication sharedApplication];  if([application canOpenURL:fallbackURL]){  [application openURL:fallbackURL];  } |

For Example, if your fallback URL is testApp://result?myAppData then Vipps will reply with “testApp://result?myAppData&status=301”

##### Registering 3rd Party app with URL Scheme and handling custom URL Calls

Defining your app's custom URL scheme is all done in the Info.plist file. Click on the last line in the file and then click the "+" sign off to the right to add a new line. Select URL Types for the new item. Once that's added, click the grey arrow next to "URL Types" to show "Item 0". Set your URL identifier to a unique string - something like com.yourcompany.yourappname.

After you've set the URL identifier, select that line and click the "+" sign again, and add a new item for URL Schemes. Then click the grey arrow next to "URL Schemes" to reveal "Item 0". Set the value for Item 0 to be your URL scheme name.



In order for your app to respond when it receives a custom URL call, you must implement the application:handleOpenURL method in the application delegate class:

|  |
| --- |
| - (BOOL)application:(UIApplication \*)application handleOpenURL:(NSURL \*)url {  *// handler your code here*  NSURLComponents \*urlComponents = [NSURLComponents componentsWithString:baseURL];    NSMutableArray <NSURLQueryItem \*>\*queryItems = urlComponents.queryItems;    *//fetch the value of a particular paramerter from queryItems array.*  } |

### Appswitch with Android platform

Following section explains how appswitch will happen for Vipps app on Android platform.

#### Overview for Android

Vipps Android platform supports two ways of appswitch integration:

* Use “startActivityForResult”

In order to use this the merchant need to set a “fallbackURL” as “INTENT”. In this way of communication there is no need to register for Url scheme.

* Use URL scheme

It is similar to the way it is solved for iOS. First the app needs to be registered for URL scheme and then pass the URL scheme in “fallbackURL”.

#### Switch from Source App to Android Vipps App

3rd party applications can integrate with Vipps by taking use of one of the following two approaches

##### Android Intent

In case of Android Intent system, in backend API call(defined later) “INTENT” should be passed in fallbackURL. And below code should be used to launch Vipps application.

|  |
| --- |
| **try** {  PackageManager pm = context.getPackageManager();  PackageInfo info = pm.getPackageInfo("no.dnb.Vipps", PackageManager.GET\_ACTIVITIES);  **if**(versionCompare(info.versionName, "1.8.0") >= **0**) {  String uri = deeplinkURL; //Use deeplink url provided in API response  Intent intent = **new** Intent(Intent.ACTION\_VIEW);  intent.setData(Uri.parse(uri));  startActivityForResult(intent,requestCode);  } **else** {  // Notify user to download the latest version of Vipps application.  }  } **catch** (PackageManager.NameNotFoundException e) {  // No Vipps app! Open play store page.  String url = " https://play.google.com/store/apps/details?id=no.dnb.Vipps";  Intent storeIntent = **new** Intent(Intent.ACTION\_VIEW);  storeIntent.setData(Uri.parse(url));  startActivity(storeIntent);  } |

##### Android URL Scheme

Following is the code sample for Android URL scheme approach.

|  |
| --- |
| **try** {  PackageManager pm = context.getPackageManager();  PackageInfo info = pm.getPackageInfo("no.dnb.Vipps", PackageManager.GET\_ACTIVITIES);  **if** (versionCompare(info.versionName, "1.4.0") >= **0**) {  String uri = deeplinkURL; //Use deeplink url provided in API response  Intent intent = **new** Intent(Intent.ACTION\_VIEW);  intent.setData(Uri.parse(uri));  startActivity(intent);  } **else** {  // Notify user to download the latest version of Vipps application.  }  } **catch** (PackageManager.NameNotFoundException e) {  // No Vipps app! Open play store page.  String url = " https://play.google.com/store/apps/details?id=no.dnb.Vipps";  Intent storeIntent = **new** Intent(Intent.ACTION\_VIEW);  storeIntent.setData(Uri.parse(url));  startActivity(storeIntent);  } |

#### Redirect Back to Source App from Android Vipps App

Android supports two ways of redirecting back to source app and merchant should use the correct method to open Vipps and get redirected back to merchant’s source app.

Below are two ways of redirecting back to source app from Android Vipps app:

##### Android Intent

Register the activity in manifest file which will handle result of Vipps response.

For Example :

|  |
| --- |
| <activity  android:name=".MainActivity"  android:label="@string/app\_name">  </activity> |

Receiving activity has to override onActivityResult method to handle result sent by Vipps application.

For Example:

|  |
| --- |
| **@Override**  **protected** **void** **onActivityResult**(**int** requestCode, **int** resultCode, Intent data) {  **if** (resultCode == RESULT\_OK) {  **if** (requestCode == **1**) {  String url = **null**;  **if** (data != **null** && data.getExtras() != **null**) {  Bundle mBundle = data.getExtras();  **if** (mBundle.get("data") != **null**) {  **try** {  url = URLDecoder.decode(mBundle.get("data").toString(), "UTF-8");  Uri parseUri = Uri.parse(url);  String status = parseUri.getQueryParameter("status");  //TODO Handle status  } **catch** (UnsupportedEncodingException e) {  e.printStackTrace();  }  }  }  }  }  } |

##### Android URL Scheme

Vipps provides a custom URL scheme to interact with Vipps. If 3rd party application wants to open Vipps application with custom URL scheme then they can implement this approach.

**Set filter in Manifest file:**

To receive a call back from the Vipps application to an activity there has to be set a filter to that activity. In the example below MainActivity is the receiving activity and Vipps application sends a response to the activity. For this activity one can set a custom URL scheme inside the intent filter.

For Example:

<activity android:name=".MainActivity"  android:label="@string/app\_name"

android:launchMode="singleInstance">

<intent-filter>

<action android:name="android.intent.action.VIEW" />

<category android:name="android.intent.category.DEFAULT" />

<category android:name="android.intent.category.BROWSABLE" />

<data android:scheme="sampleApps" />

</intent-filter>

</activity>

**Note:** scheme should be same as you send in fallbackURL parameter in Vipps API

Vipps application will send the result to the 3rd party application by starting a new activity with the fallbackURL as a URI parameter in the intent. The 3rd party application can make their receiving activity as a *singleInstance* to handle the response in same activity.

The receiving activity has to override onNewIntent method to handle result send by Vipps application.

|  |
| --- |
| **@Override**  **protected** **void** **onNewIntent**(Intent intent) {  **super**.onNewIntent(intent);  String url = **null**;  **if** (intent != **null** && intent.getData() != **null**) {  **try**{  url = URLDecoder.decode(intent.getData().toString(),"UTF-8");  Uri parseUri = Uri.parse(url);  String status = parseUri.getQueryParameter("status");  //TODO Handle status  }**catch**(UnsupportedEncodingException e) {  e.printStackTrace();  }  }  } |

## List of error codes for deeplinking

Following are the identified status codes merchant may receive from Vipps app.

|  |  |
| --- | --- |
| Status Code | Description |
| 100 | Success |
| 302 | User doesn’t have Vipps profile |
| 303 | Login failed (login max attempt reached) |
| 304 | Vipps doesn’t support this action, please update Vipps |
| 401 | Request timed out or Token has expired |
| 451 | The user was selected for fraud validation |
| 999 | Failed |

Below are the status code ranges which Vipps maintains for future purposes. For example, if there is new error message related to fraud, then it will fall under range 450 to 499.

1XX – Success Scenarios

200 to 250 – Input Error

250 to 299 - User Actions

3XX – Authentication / User Profile / Merchant Profile / Configuration related error

400 to 450 – Transaction related error

450 to 499 – Fraud related error

5XX – Reserved for future use

6XX – Reserved for future use

7XX – Reserved for future use

8XX – Reserved for future use

9XX – Others

# API definitions

Below section explains different API definitions supported by Vipps ecommerce APIs.

## Request Headers

|  |  |  |  |
| --- | --- | --- | --- |
| Header Name | Header Value | Optional | Description |
| Authorization | JWT Access Token <<value>> | No | type: Authorization token  value: Access token is obtained by registering merchant backend application in Merchant Developer Portal. |
| Accept-Language | no | Yes | Allowed languages at present is Norwegian |
| Content-Type | application/json | No | Type of the body |
| X-TimeStamp | Time stamp when the request called | No | Time to call |
| X-Request-Id | To identify the idempotent request | Yes | For Making request to be idempotent this ID is must so that the system will not do any side effects.  1. Applicable to Initiate, Capture, Refund payment  2. Size should be 30  3. 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. |
| Ocp-Apim-Subscription-Key | Base 64 encoded string | No | Subscription key for eCommerce product. This can be found in User Profile page on Merchant developer portal |

## Payment sequence

Below is payment sequence diagram to understand steps to follow for using various Vipps ecommerce APIs.



## Initiate Payment

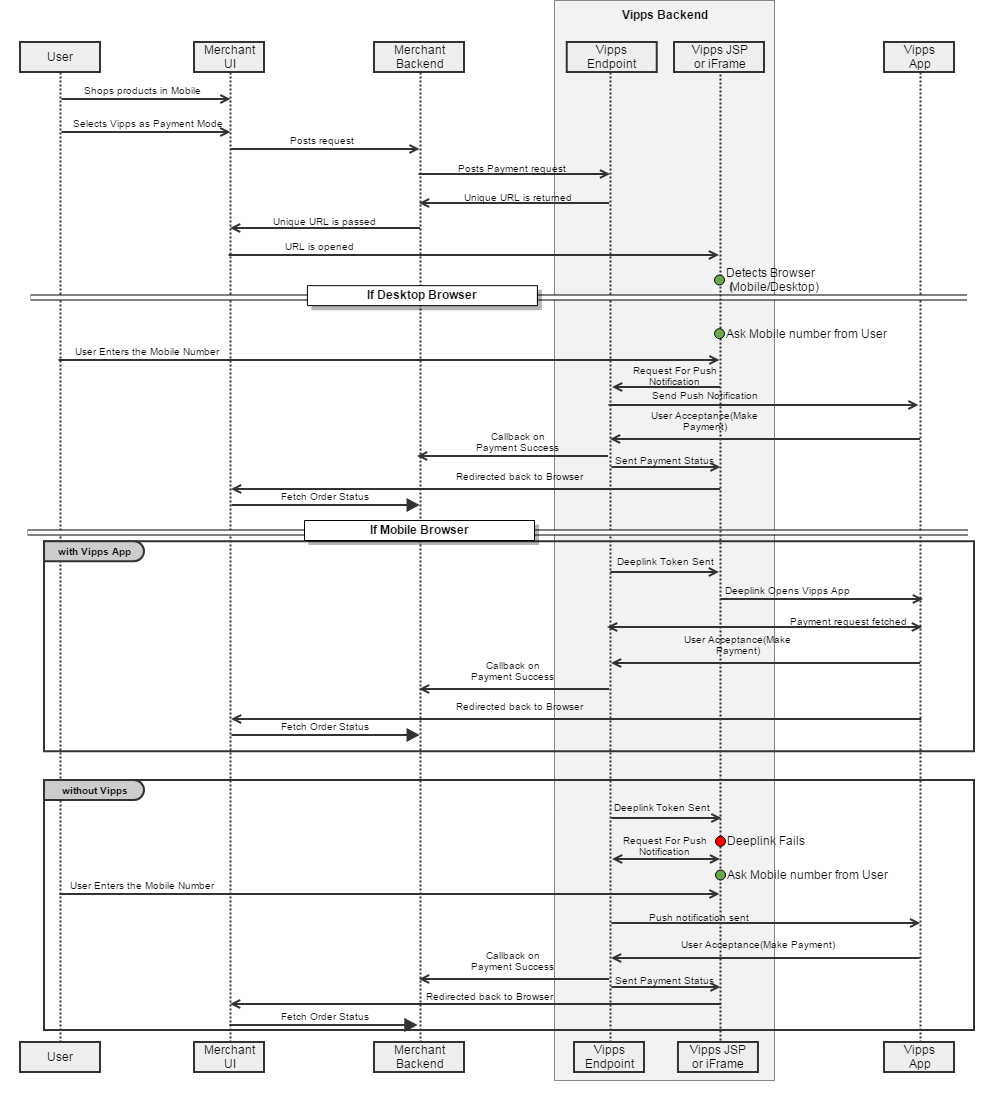
### Overview

The API Call allows merchant to initiate a payment flow by using Vipps.

In order to identify sales channel payments are coming from, a *merchantSerialNumber* is used to distinguish between them. Please note that single payment is uniquely identified by composite of *merchantApplicationId*, *merchantSerialNumber* and *orderId*.

Merchant provided *orderId* must be unique per sales channel.

Once successfully initiated the transaction in Vipps, it will give you the redirect URL as response which has to be used by merchant to open Vipps landing page. Landing page will own functionality to identify and differentiate request coming from mobile browser/desktop browser.

**Payment triggered from desktop and mobile browser** 

Flow when user initiates payment from mobile browser where Vipps is present in same device –

1. Landing page will check if Vipps app is available in same mobile device.
2. If Vipps app is available then it will invoke Vipps app and landing page will be closed.
3. Here after based on user interactions (accept / reject) further payment steps will be followed.
4. Once payment process is completed, Vipps will call fallback URL to redirect to original mobile browser page.
5. If merchant do not receive callback from Vipps, then they have to confirm their order status from Vipps by calling getOrderStatus service.

Flow when user initiates payment from mobile browser where Vipps is NOT present in same device –

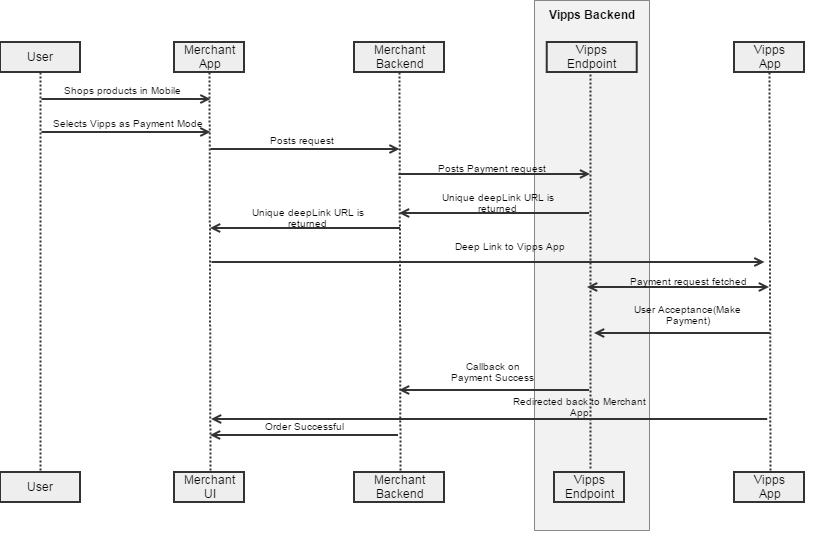
1. Landing page will check if Vipps app is available in same mobile device.
2. If Vipps app is not available then landing page will ask for user’s mobile number. After user enters mobile number, Vipps will send push notification to corresponding Vipps profile, if exists. Landing page is not closed in this case.
3. Vipps user needs to accept/reject the payment request coming from merchant for further payment steps.
4. Once payment process is completed, landing page will redirect to mobile web browser where payment was initiated.
5. If merchant do not receive callback from Vipps, then they have to confirm their order status from Vipps by calling getOrderStatus service.

Flow when user initiates payment from desktop browser –

1. Landing page will be opened inside desktop browser.
2. As Vipps app is not available in the vicinity, landing page will ask for user’s mobile number. After user enters mobile number, Vipps will send push notification to corresponding Vipps profile, if exists. Landing page is not closed in this case.
3. Vipps user needs to accept/reject the payment request coming from merchant for further payment steps.
4. Once payment process is completed, landing page will redirect to desktop web browser where payment was initiated.
5. If merchant do not receive callback from Vipps, then they have to confirm their order status from Vipps by calling getOrderStatus service.

**Payment triggered from Merchant Mobile App**

Vipps will identify the request coming from mobile app of merchant from initiate request body parameter (**isApp**). In this case, Vipps backend will send the URI which merchant should use to invoke Vipps app directly. Landing page is not involved in this case.



1. Vipps will identify the request coming from merchant mobile app based on isApp parameter value.
2. If value is true then Vipps will send deeplink URI as response to initiate payment.
3. Merchant as to use this URI to invoke Vipps app for user to proceed with payment.
4. Vipps user needs to accept/reject the payment request coming from merchant for further payment steps.
5. Once payment process is completed, Vipps app will redirect to merchant mobile app where payment was initiated.
6. If merchant do not receive callback from Vipps, then they have to confirm their order status from Vipps by calling getOrderStatus service.

When user confirms the payment in Vipps app -

After the customer has confirmed payment, Vipps will execute funds reservation on customer card used in transaction in order to secure future capture. Please note that in a case of direct capture, reservation and capture is done in a single step.

If the funds reservation fails for any reason (communication error, credit card expired, not enough funds to reserve) Vipps will cancel the payment flow and inform the merchant about outcome. Merchant’s *orderId* used for cancelled payment flow cannot be reused for a new initiate payment service call.

*merchantSerialNumber* are provided to merchant by Vipps after merchant account is created or new sales channel is enrolled.

### URL for Initiate Payment

/v2/payments

### Method

POST

### ****Request Body****

|  |
| --- |
| {      "customerInfo": {          "mobileNumber": "90090900"      },      "merchantInfo": {  “authToken” : [String} 255,          "callbackPrefix": "https://www.domain.no/ecomm", [String] REQUIRED  “consentRemovalPrefix” : "https://www.domain.no/ecomm", [String] OPTIONAL  “fallBack”: “https:// dnsname/contextPath /api/fallback?sessionId:1234566”,  “isApp” : “True/False” , Boolean DEFAULT FALSE          "merchantSerialNumber": "NSBWSHP12",  “paymentType”: “eComm Express Payment/eComm Regular Payment”,  “shippingDetailsPrefix” : "https://www.domain.no/ecomm" [String] OPTIONAL      },      "transaction": {          "amount": 1200,          "orderId": "219930212",  "refOrderId": "119930211",          "timeStamp":"2014-06-24T08:34:25-07:00",          "transactionText": "Transaction text"      }  } |

### ****Request Description****

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | Type | Size | | Optional | | Description |
| merchantSerialNumber | String | 6 | No | | Identifies a merchant sales channel i.e. website, mobile app etc. | |
| callbackPrefix | String | 255 | No | | This is to receive the callback after the payment request. Domain name and context path should be provided by merchant as the value for this parameter. The rest of the URL will be appended by Vipps according to Vipps guidelines. | |
| shippingDetailsPrefix | String | 255 | Yes | | In case of express checkout payment, merchant should pass this prefix to let Vipps fetch shipping cost and method related details | |
| consentRemovalPrefix | String | 255 | Yes | | In case of expess checkout payments, this callback will be used for informing merchant about consent removal from Vipps user. This means that particular user do not want merchant to store/use his personal information anymore. | |
| fallBack | String | 255 | No | | Vipps will use the fall back URL to redirect Merchant Page once Payment is completed in Vipps System | |
| isApp | Boolean | 255 | No | | This parameter indicates whether payment request is triggered from Mobile App or Web browser. Based on this value, response will be redirect url for Vipps landing page or deeplink Url to connect vipps App | |
| authToken | String | 255 | Yes | | Merchant should share this token if merchant has authentication mechanism in place which could be used for making callbacks secure | |
| mobileNumber | String | 8 | Optional | | Mobile number of the user who has to pay for the transaction from Vipps.  Allowed format:  xxxxxxxx | |
| orderId | String | 30 | No | | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters | |
| refOrderId | String | 30 | Yes | | Identifies if the payment references to some past orders registered with Vipps. If defined, transactions for this payment will show up as child transactions of the specified order. | |
| Amount | Integer | - | No | | Amount in øre. 32 Bit Integer (2147483647) | |
| transactionText | String | 100 | No | | Transaction text that can be displayed to end user | |
| timeStamp | String | - | Yes | | Timestamp in ISO-8601 representing when the order has been made by merchant | |
| paymentType | String | - | No | | This will parameter will identify difference between ecomm payment and ecomm express payment. | |

### Success Response

|  |  |
| --- | --- |
| Http Status Code | Content |
| 200 | ok |

### ****Response Body****

|  |
| --- |
| {      "orderId": "219930212",      “url”:<https://vipps.no/payments/paymentgateway.jsp?token=fasdfqw04rtasdkfam.asdfqw30rsdfasd802354.asdfajsjq340sfa> or Vipps://app?token= fasdfqw04rtasdkfam.asdfqw30rsdfasd802354.asdfajsjq340sfa”        }  } |

### ****Description****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Type | Size | Optional | Description |
| orderId | String | 30 | No | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters |
| url | String | 255 | No | Url parameter will have url to redirect the request to vipps gateway page in case request is trigger from web browser or deeplink url to open vipps app incase request is triggered from Merchant Mobile App |

### Error Response

|  |  |  |
| --- | --- | --- |
| Http Status Code | Content | Description |
| 401 | Unauthorized | API call authorization failed |
| 400 | Bad request | API request validation failed because of malformed request object. More details will be provided by the error object |
| 403 | Forbidden | The request has been understood but failed because of some reason |
| 5xx | Internal server error | Internal server error |

## Cancel Payment

### ****Overview****

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.

### URL

/v2/payments/{orderId}/cancel

### Method

PUT

### URLParams

orderId =[Integer | String] REQUIRED

### ****Request Body****

|  |
| --- |
| {      "merchantInfo": {          "merchantSerialNumber": "NSBWSHP12"      },      "transaction": {          "transactionText":"transaction text"      }  } |

### ****Description****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Type | Size | Optional | Description |
| merchantSerialNumber | String | 6 | No | Identifies a merchant sales channel i.e. website, mobile app etc. |
| transactionText | String | 100 | No | Reference text for the merchant |

### Success Response

|  |  |
| --- | --- |
| Http Status Code | Content |
| 200 | ok |

### ****Response Body****

|  |
| --- |
| {      "orderId": "219930212",      "transactionInfo": {          "amount": 1200,          "status": "Cancelled",          "timeStamp": "2014-06-24T08:34:25-07:00",          "transactionId": "100025255",          "transactionText": "Refrence text"      },      "transactionSummary": {          "capturedAmount":0,          "refundedAmount":0,          "remainingAmountTocapture":0,          "remainingAmountToRefund":0      }  } |

### ****Description****

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Type | Optional | Description |
| orderId | String | No | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters |
| amount | Integer | No | Ordered amount in øre |
| timeStamp | String | No | Timestamp in ISO-8601 representing when vipps Cancelled transaction. |
| transactionText | String | No | Transaction text reference provided by merchant |
| status | String | No | Status of the ordered transaction |
| transactionId | String | No | Vipps transaction id |
| capturedAmount | Integer | No | Total amount captured |
| remainingAmountTocapture | Integer | No | Total remaining amount to capture |
| refundedAmount | Integer | No | Total refunded amount of the order |
| remainingAmountToRefund | Integer | No | Total remaining amount to refund |

### Error Response

|  |  |  |
| --- | --- | --- |
| Http Status Code | Content | Description |
| 401 | Unauthorized | API call authorization failed |
| 400 | Bad request | API request validation failed because of malformed request object. More details will be provided by the error object |
| 402 | Payment Cancellation Failed | The request has been understood but payment failed because of unable to cancel the reservation |
| 403 | Forbidden | The request has been understood but failed because of reason detailed out in the error message |
| 5xx | Internal server error | Internal server error |

## Capture Payment

### ****Overview****

The API call allows merchant to capture the reserved amount. Amount to capture cannot be higher than reserved. The API also allows capturing partial amount of the reserved amount. Partial capture can be called as many times as required so long there is reserved amount to capture. Transaction text is not optional and is used as a proof of delivery (tracking code, consignment number etc.).

In a case of direct capture, both fund reservation and capture are executed in a single operation.

### URL

/v2/payments/{orderId}/capture

### Method

POST

### URLParams

orderId =[Integer | String] REQUIRED

### ****Request Body****

|  |
| --- |
| {  "merchantInfo": {          "merchantSerialNumber": "NSBWSHP12"      },      "transaction": {          "amount": 100,          "transactionText":"transaction text "      }  } |

### ****Description****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Type | Size | Optional | Description |
| merchantSerialNumber | String | 6 | No | Identifies a merchant sales channel i.e. website, mobile app etc. |
| orderId | String | 30 | No | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters |
| amount | Integer | - | Yes | Amount in øre, if amount is 0 or not provided then full capture will be performed. 32 Bit Integer (2147483647) |
| transactionText | String | 100 | No | Proof of delivery |

### Success Response

|  |  |
| --- | --- |
| Http Status Code | Content |
| 200 | ok |

### ****Response Body****

|  |
| --- |
| {      "orderId": "219930212",      "transactionInfo": {          "amount": 1200,          "status": "Capture",          "timeStamp": "2014-06-24T08:34:25-07:00",          "transactionId": "10001234567",          "transactionText": "Refrence text"      },      "transactionSummary": {          "capturedAmount":"100",          "refundedAmount":"0",          "remainingAmountToCapture":"1100",          "remainingAmountToRefund":"100"      }  } |

### ****Description****

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Type | Optional | Description |
| orderId | String | No | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters |
| amount | Integer | No | Ordered amount in øre |
| timeStamp | String | No | Timestamp in ISO-8601 representing when vipps Captured transaction. |
| transactionText | String | No | Transaction text reference provided by merchant |
| status | String | No | Status of the ordered transaction |
| transactionId | String | No | Vipps transaction id |
| capturedAmount | Integer | No | Total amount captured |
| remainingAmountTocapture | Integer | No | Total remaining amount to capture |
| refundedAmount | Integer | No | Total refunded amount of the order |
| remainingAmountToRefund | Integer | No | Total remaining amount to refund |

### Error Response

|  |  |  |
| --- | --- | --- |
| Http Status Code | Content | Description |
| 401 | Unauthorized | API call authorization failed |
| 400 | Bad request | API request validation failed because of malformed request object. More details will be provided by the error object |
| 402 | Payment Failed | The request has been understood but payment failed because of issuer bank |
| 403 | Forbidden | The request has been understood but failed because of some reason |
| 5xx | Internal server error | Internal server error |

## Refund Payment

### ****Overview****

The API allows a merchant to do a refund of already captured transaction. There is an option to do a partial refund of the captured amount. Refunded amount cannot be larger than captured.

Timeframe for issuing a refund for a payment is 365 days from the date payment has been captured. If the refund payment service call is called after the refund timeframe, service call will respond with an error.

Refunded funds will be transferred from the merchant account to the customer credit card that was used in payment flow. Pay attention that in order to perform refund, there must be enough fund at merchant settlements account.

### URL

/v2/payments/{orderId}/refund

### Method

POST

### URLParams

orderId =[Integer | String] REQUIRED

### ****Request Body****

|  |
| --- |
| {  "merchantInfo": {          "merchantSerialNumber": "NSBWSHP12"      },      "transaction": {          "amount":123,          "transactionText":"Transaction text"      }  } |

### ****Description****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Type | Size | Optional | Description |
| merchantSerialNumber | String | 6 | No | Identifies a merchant sales channel i.e. website, mobile app etc. |
| orderId | String | 30 | No | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters |
| amount | Integer | - | Yes | Amount in øre, if amount is 0 or not provided then full refund will be performed. 32 Bit Integer (2147483647) |
| transactionText | String | 100 | No | Proof of delivery |

### Success Response

|  |  |
| --- | --- |
| Http Status Code | Content |
| 200 | ok |

### ****Response Body****

|  |
| --- |
| {      "orderId": "219930212",      "transaction": {          "amount": 1200,          "status": "Refund",          "timeStamp": "2014-06-24T08:34:25-07:00",          "transactionId": "100023434",          "transactionText": "Refrence text"      },      "transactionSummary": {          "capturedAmount":"0",          "refundedAmount":"1200",          "remainingAmountToCapture":"0",          "remainingAmountToRefund":"0"      }  } |

### ****Description****

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Type | Optional | Description |
| orderId | String | No | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters |
| amount | Integer | No | Ordered amount in øre |
| timeStamp | String | No | Timestamp in ISO-8601 representing when vipps did the requested operation |
| transactionText | String | No | Transaction text reference provided by merchant |
| status | String | No | Status of the ordered transaction |
| transactionId | String | No | Vipps transaction id |
| capturedAmount | Integer | No | Total amount captured |
| remainingAmountToCapture | Integer | No | Total remaining amount to capture |
| refundedAmount | Integer | No | Total refunded amount of the order |
| remainingAmountToRefund | Integer | No | Total remaining amount to refund |

### Error Response

|  |  |  |  |
| --- | --- | --- | --- |
| Http Status Code | | Content | Description |
| 401 | Unauthorized | | API call authorization failed |
| 400 | Bad request | | API request validation failed because of malformed request object. More details will be provided by the error object |
| 402 | Payment Failed | | The request has been understood but payment failed because of issuer bank |
| 403 | Forbidden | | The request has been understood but failed because of some reason |
| 5xx | Internal server error | | Internal server error |

## Get Payment Details

### ****Overview****

The API allows merchant to get the details of a payment transaction. Service call returns detailed transaction history of given payment where events are sorted by the time single transaction occurred.

### URL

v2/payments/{orderId}/details

### Method

GET

### URLParams

orderId =[Integer | String] REQUIRED

### Success Response

|  |  |
| --- | --- |
| Http Status Code | Content |
| 200 | ok |

### ****Response Body****

|  |
| --- |
| {     "orderId": "219930212",  "shippingDetails" : {              "address" : {                  "addressLine1":"",[String] REQUIRED                  "addressLine2":"",[String] OPTIONAL                  "city":"",[String] REQUIRED                  "country" : "",[String] REQUIRED Default NO                  "postCode":""[String] REQUIRED              },              "shippingCost" : 50.89, [BigDecimal] REQUIRED Scale 2              "shippingMethod" : ""[String] REQUIRED              "shippingMethodId" : ""[String] REQUIRED      } ,      "transactionLogHistory": [{          "amount": "",          "operation": "",          "operationSuccess": "",          "requestId": ""          "timeStamp": "",          "transactionId":"",          "transactionText":""      }],      "transactionSummary": {          "capturedAmount":"0",          "refundedAmount":"1200",          "remainingAmountTocapture":"0",          "remainingAmountToRefund":"0"      },    "userDetails" : {          "bankIdVerified" : "Y",[Char] OPTIONAL Y/N only          "dateOfBirth" : "",[String] OPTIONAL          "email" : "",[String] REQUIRED          "firstName" : "",[String] REQUIRED          "lastName" : "",[String] REQUIRED          "mobileNumber" : "",[String] REQUIRED Length=8          "ssn" : "",[String] OPTIONAL Length=11  "userId" : ""[String] REQUIRED      }  } |
|  |

### ****Description – Transaction Summary****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | | Type | Optional | Description |
| orderId | String | No | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters |
| capturedAmount | Integer | No | Total amount captured |
| remainingAmountTocapture | Integer | No | Total remaining amount to capture |
| refundedAmount | Integer | No | Total refunded amount of the order |
| remainingAmountToRefund | Integer | No | Total remaining amount to refund |

### ****Description – Transaction Log History****

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Type | Optional | Description |
| timeStamp | String | No | Timestamp in ISO-8601 representing when vipps did the requested operation |
| operation | String | No | Log for the operation |
| operationSuccess | Boolean | No | Success or failure for the given operation |
| amount | Integer | No | Amount performed on the given operation |
| transactionId | String | No | Vipps Transaction Id for the operation |
| transactionText | String | Yes | Reference text provided by the merchant during the operation |
| requestId | String | Yes | Idempotent request id provided for the operation |

### ****Description – Shipping Details****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | | Type | Optional | Description |
| shippingDetails | String | Yes | Will contain shipping details for an order. |
| userId | String | Yes | This will uniquely identify a user in Vipps and merchant system. Merchant is required to store this field for future references. |
| userDetails | String | Yes | User Details in which few fields are optional. Merchant needs to ask for SSN, dateOfBirth and isBankIdVerified explicitly during onboarding. |

### Error Response

|  |  |  |
| --- | --- | --- |
| Http Status Code | Content | Description |
| 401 | Unauthorized | API call authorization failed |
| 400 | Bad request | API request validation failed because of malformed request object. More details will be provided by the error object |
| 404 | Resource Not Found | The request has been understood but payment failed because of issuer bank |
| 403 | Forbidden | The request has been understood but failed because of some reason |
| 5xx | Internal server error | Internal server error |

## Get Order Status

### ****Overview****

The API call allows merchant to get the status of the last payment transaction. Primarily use of this service is meant for inApp where simple response to check order status is preferred. The call doesn’t give any transaction history, but only a simple status of the last payment transaction.

### URL

V2/payments/{orderId}/status

### Method

GET

### URLParams

orderId =[Integer | String] REQUIRED

### Success Response

|  |  |
| --- | --- |
| Http Status Code | Content |
| 200 | ok |

### ****Response Body****

|  |
| --- |
| {      "orderId": "219930212",      "transactionInfo": {          "amount": 1200,          "status": "FAILED",          "timeStamp": "2014-06-24T08:34:25-07:00",          "transactionId": "100023434"      }  } |

### ****Description****

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Type | Optional | Description |
| orderId | String | No | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters |
| Amount | Integer | No | Payment amount |
| transactionId | String | No | Vipps Transaction Id for the payment |
| timestamp | String | No | Timestamp in ISO-8601 representing when vipps did the requested operation |
| status | Enum | No | State of the parent transactions. Child transaction state like Capture\Refund will not be captured here. Potential statuses are described in section 5.7 |

## Fetch Shipping Cost & Method (Hosted by Merchant for express checkout)

### Overview

This API call allows Vipps to get the shipping cost and method based on the provided address and product details. Primarily use of this service is meant for ecomm express checkout where Vipps needs to present shipping cost and method to the vipps user. This service is to be implemented by merchants.

### Request Header

|  |  |  |  |
| --- | --- | --- | --- |
| Header Name | Type | Optional | Description |
| Authorization | String | Yes | type: Authorization token  value: merchant’s authorization token for secure callbacks |

### URL

[shippingDetailsPrefix]/v2/payments/{orderId}/shippingDetails

### Method

POST

### URLParams

orderId =[String] REQUIRED

### Success Response

|  |  |
| --- | --- |
| Http Status Code | Content |
| 200 | Ok |

### Request Body

|  |
| --- |
| {  "addressId": 178687, [Integer] REQUIRED      "addressLine1": "Fernanda nissens Gate 10B",[String] REQUIRED  "addressLine2": "",[String] REQUIRED  "city": "Oslo",[String] REQUIRED  "country": "NO",[String] REQUIRED only NO  "postCode": 0484[String] REQUIRED length=4  } |

### Response Body

|  |
| --- |
| {      "addressId": 178687 [Integer] REQUIRED,  "orderId" : “” [String] REQUIRED,      "shippingDetails": [{          "isDefault": "Y" , [String] REQUIRED Only Y/N          "priority": 1, [Integer] OPTIONAL          "shippingCost": "40.89",[BigDecimal] REQUIRED Scale 2          "shippingMethod": ""[String] REQUIRED          "shippingMethodId": ""[String] REQUIRED      }]  } |

### Request Description

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Type | Optional | Description |
| orderId | String | No | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters |
| addressLine1 | String | No | Free Text |
| addressLine2 | String | Yes | Free Text |
| City | String | No | Free Text |
| postCode | Integer | No | 4 Digit |
| Country | String | No | “NO” Only country supported is Norway |
| addressId | Integer | No | Vipps Provided address Id. To be returned in response in the same field |

### Response Description

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Type | Optional | Description |
| orderId | String | No | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters |
| addressId | Integer | No | Vipps Provided address Id |
| shippingCost | BigDecimal | No | Payment amount with scale 2 and “.” as decimal separator. This field should be corresponding to order id, address and shipping method. |
| isDefault | Char | No | Only one shipping method should be default. Possible values Y or N |
| Priority | Integer | Yes | Sequence in which shipping method to be displayed to the Vipps user during express checkout. |
| shippingMethod | String | No | Free text. Example :  PICKUP AT POSTEN  DELIVERY IN 5-7 DAYS  EXPRESS DELIVERY IN 1 DAY |

## Callback : Transaction Update

### Overview

This API allows Vipps to send the transaction details.

During regular ecomm payment order and transaction details will be shared.

During express checkout payment it will provide user details and shipping details addition to the order and transaction details.

### Request Header

|  |  |  |  |
| --- | --- | --- | --- |
| Header Name | Type | Optional | Description |
| Authorization | String | Yes | type: Authorization token  value: merchant’s authorization token for secure callbacks |

### URL

[callbackPrefix]/v2/payments/{orderId}

### Method

POST

### URLParams

orderId =[String] REQUIRED

### Success Response

|  |  |
| --- | --- |
| Http Status Code | Content |
| 200 | Ok |

### Request Body

Regular ecomm payment

|  |
| --- |
| {      "orderId": "219930212",[String] REQUIRED      "transactionInfo": {          "amount": 120000, [Integer] REQUIRED Scale 2          "timeStamp": "2014-06-24T08:34:25-07:00",          "status": "Reserve",          "transactionId": "1000234732"      },      "errorInfo":{          "errorCode": "",          "errorGroup":"",          "errorMessage": ""      }} |

Express Payment Example

|  |
| --- |
| {  "merchantSerialNumber": "csdac33",[String] REQUIRED      "orderId": "219930212",[String] REQUIRED      "shippingDetails" : {              "address" : {                  "addressLine1":"",[String] REQUIRED                  "addressLine2":"",[String] OPTIONAL                  "city":"",[String] REQUIRED                  "country" : "",[String] REQUIRED Default NO                  "postCode":""[String] REQUIRED              },              "shippingCost" : 50.89, [BigDecimal] REQUIRED Scale 2              "shippingMethod" : ""[String] REQUIRED              "shippingMethodId" : ""[String] REQUIRED      } ,      "transactionInfo": {          "amount": 1200, [Integer] REQUIRED Scale 2          "status": "Reserve",          "timeStamp": "2014-06-24T08:34:25-07:00",          "transactionId": "1000234732"      },    "userDetails" : {          "bankIdVerified" : "Y",[Char] OPTIONAL Y/N only          "dateOfBirth" : "",[String] OPTIONAL          "email" : "",[String] REQUIRED          "firstName" : "",[String] REQUIRED          "lastName" : "",[String] REQUIRED          "mobileNumber" : "",[String] REQUIRED Length=8          "ssn" : "",[String] OPTIONAL Length=11  "userId" : ""[String] REQUIRED      }      "errorInfo":{          "errorCode": "",          "errorGroup":"",          "errorMessage": ""      },  } |

### Response Body

NA

### Request Description

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Type | Optional | Description |
| orderId | String | No | Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters |
| shippingDetails | String | Yes | Shipping details are optional and will not be sent in regular ecomm payment.  Shipping details are mandatory for Express  checkout payment. |
| userDetails | String | Yes | User details are optional and will not be sent in regular ecomm payment.  User details are mandatory for Express  checkout payment.Few fields are optional in user details. Merchant needs to ask for SSN, dateOfBirth and isBankIdVerified information explicitly during onboarding. |
| userId | String | No | This will uniquely identify a user in Vipps and merchant system. Merchant is required to store this field for future references. |

## Remove User Consent (for express checkout)

### Overview

This API allows Vipps to send consent removal request to merchant. After this merchant is obliged to remove the user details from merchant system permanently, as per the GDPR guidelines.

### URL

{consetRemovalPrefix}/v2/consents/{userId}

### Method

DELETE

### URLParams

userId =[String] REQUIRED

### Success Response

|  |  |
| --- | --- |
| Http Status Code | Content |
| 200 | ok |

### Request Body

NA

### Response Body

NA

### Request Description

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Type | Optional | Description |
| userId | String | No | This will identify a user uniquely in merchant and Vipps system.  UserId in the url is in encrypted format. So it will have some special charecters. So vipps will encode with UTF-8 in the url.Merchant should decode with UTF-8 before to find the user in their system |

# Appendix

How to use developer portal can be understood from attached document below.

