Vipps eCommerce APIs v2

Document version: **2.1**
Date produced: **Feb-2018**

Content

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.

1. Use case scenarios

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

1.1 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. 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

1.2 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.

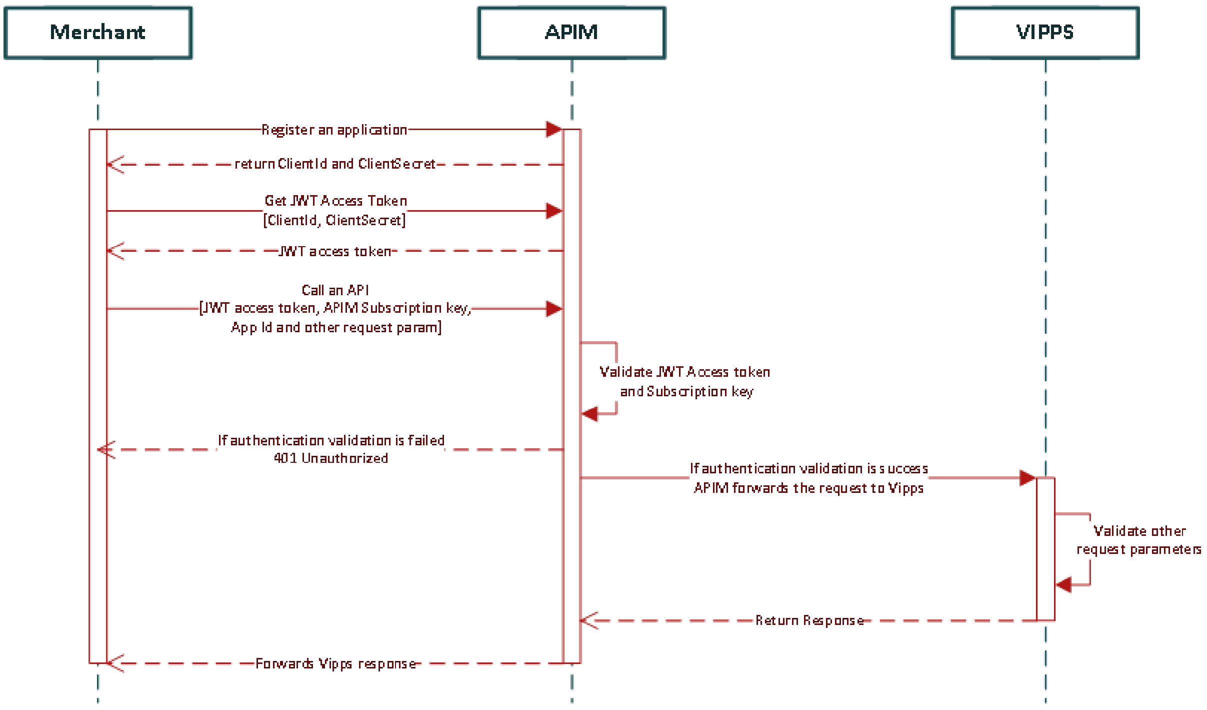
Get shipping cost & method – Used for fetching shipping cost & method combination based on user selected address
Transaction update with user details – separate callback to merchant for receiving post payment information
Remove user consent – Used to inform merchant when vipps user removes consent to share his details.

2. 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). Once logged in to the developer portal merchant finds the API credentials needed to make API calls.

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


Merchant
APIM
VIPPS
Register an application
return ClientId and ClientSecret
Get JWT Access Token [ClientId, ClientSecret]
JWT access token
Call an API [JWT access token, APIM Subscription key, App Id and other request param]
Validate JWT Access token and Subscription key
If authentication validation is failed 401 Unauthorized
If authentication validation is success APIM forwards the request to Vipps
Validate other request parameters
Return Response
Forwards Vipps response

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 receives API 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 and APIM subscription key 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.

3. Authentication

3.1 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). 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

3.2 Access Token

3.2.1 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.

3.2.2 URL

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

3.2.3 Method

POST

3.2.4 Request Headers

```
"client_id":<ClientID>
"client_secret":<ClientSecret>
"Ocp-Apim-Subscription-Key":<Ocp-Apim-Subscription-Key>
```

3.2.5 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

3.2.6 Success Response

Http Status Code	Content
200	OK

3.2.7 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":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6InowMzl6ZHNGdWl6cEJmQlZLMVRuMjVRSFlPMCIsImtpZCI6Inc
NWNiNDcxNmJjNGIvIiwiaWF0IjoxNDk1MTg0NTc0LCJuYmYiOjE0OTUxODQ1NzQsImV4cCI6MTQ5NTI3MTI3MywiYW
2NTI2LTUxZGMtNGMxNC1iMDg2LWE1Y2I0NzE2YmM0Yi8iLCJ0aWQiOiJlNTExNjUyNi01MWRjLTRjMTQtYjA4Ni1hN
_CnMRo3bXavz3Sdo2-1amFKsOY8AFODpqJR0MYqPK_Kr6sSIWL3M_L3wu0rG976HIXllsRLvWBSwDeMgBAUvwW
}
```

3.2.8 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

3.2.9 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

3.2.10 Error Response Body

3.2.10.1 400 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"
}
```

3.2.10.2 401 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"
}
```

4. Idempotency

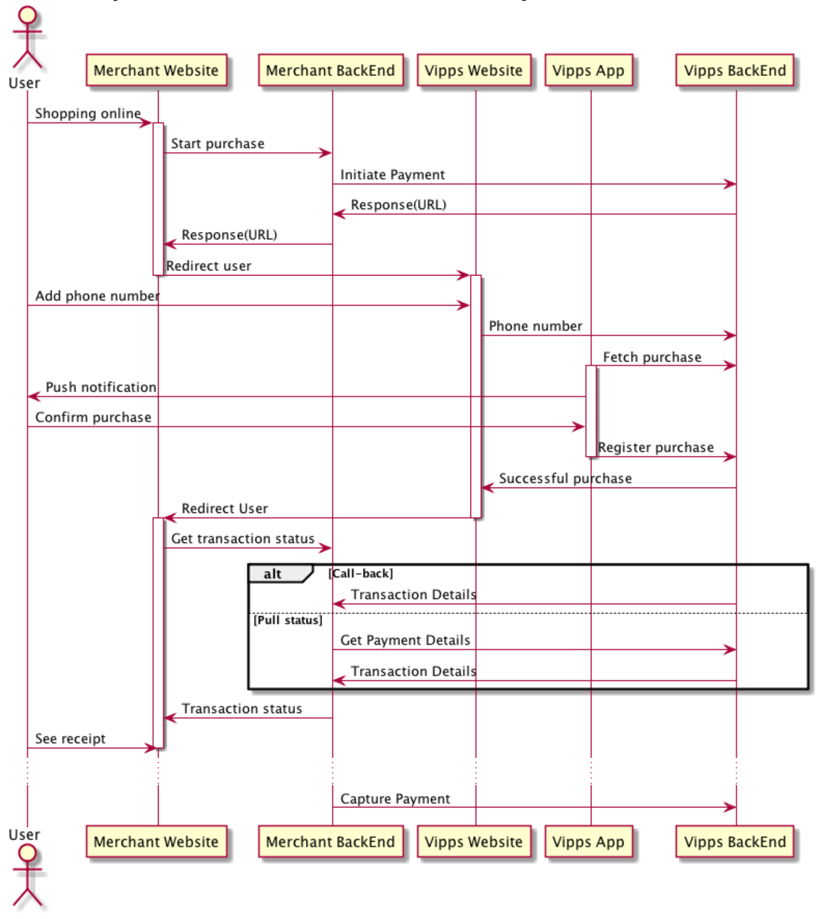
All API requests in Vipps eCommerce can be retried without any side effects by providing idempotent key in a header of the request. For example, in case the request fails because of network error it can safely be retried with the same idempotent key. Idempotent key is generated by merchant.

```
-H "X-Request-Id: slvnwdcweofjwefweklfwelf"
```

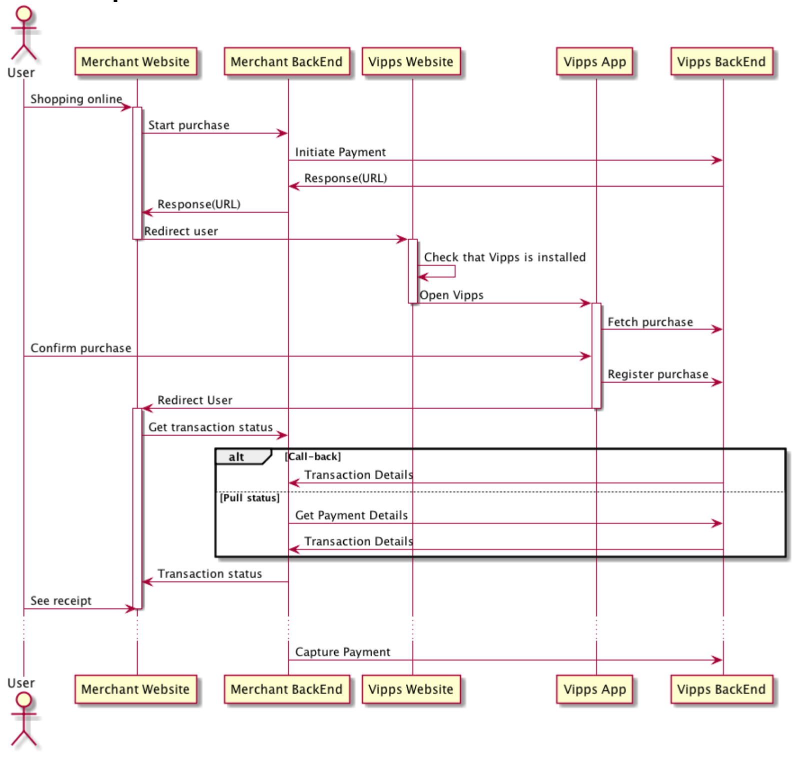
5. eCommerce Payment Flows

Payment flow in Vipps eCommerce is represented by following diagram:

Flow for push notification flow on desktop browser:



Flow for push notification flow on mobile:



5.1 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.

5.2 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.

5.3 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.

5.4 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.

5.5 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.

5.6 Refund

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

5.7 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

5.8 Get Payment Details

Get Payment Details will provide all the payment transaction details.

6. 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.

6.1 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.

6.2 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).

6.3 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.

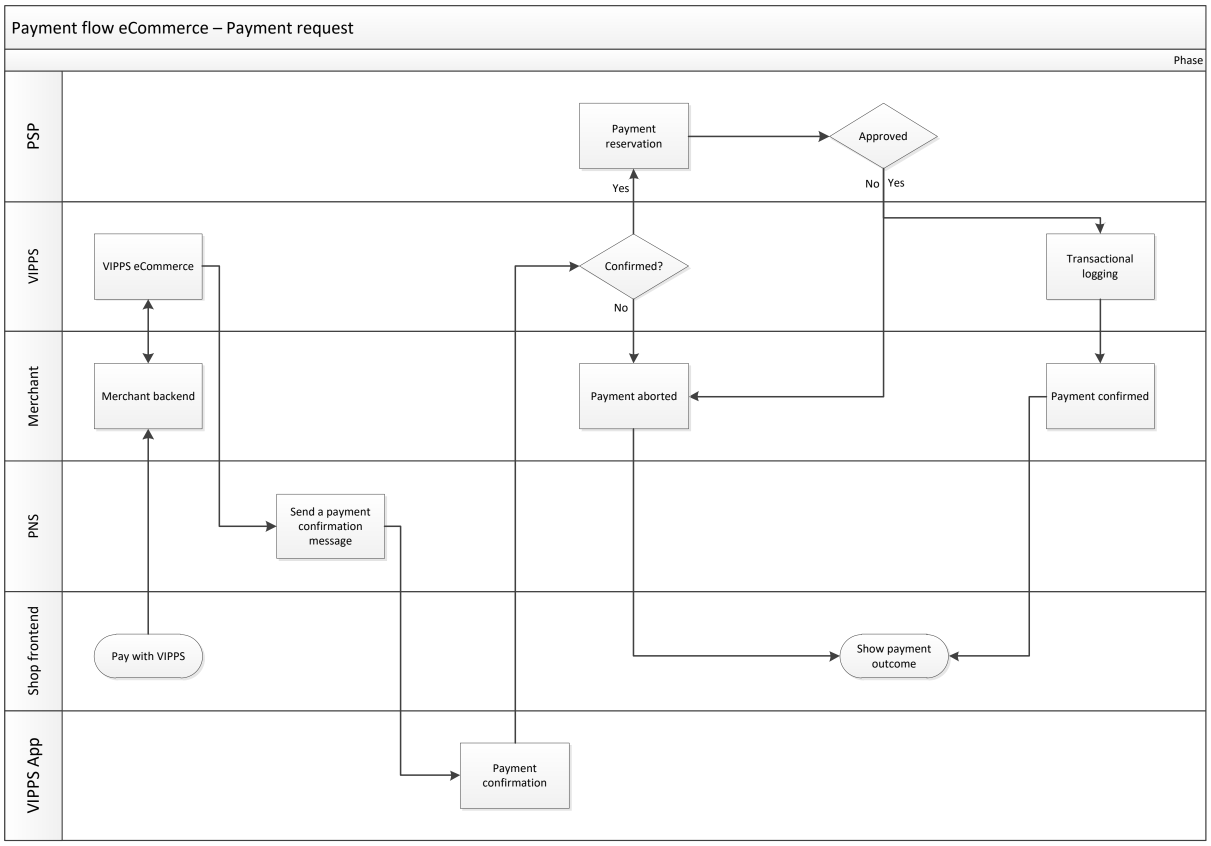
7. Exception handling

7.1 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.

7.2 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.

7.2.1 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 call Get Payment Details and check status of last transaction in transaction history prior executing the service call again.

7.2.2 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 not be retried.
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.

7.2.3 PSP connection issues

In a case when Vipps experiences communication problems with PSP, service call will respond with 402 HTTP Error. Merchant should make a call to Get Payment Details to check if the transaction request is processed before making service call (with same idempotency key) again.

8. 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.

8.1 Success Codes

200 – OK
202 - Accepted

8.2 HTTP 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

8.3 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

8.4 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

9. 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.

9.1 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).
- 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".

9.2 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.

9.2.1 Appswitch with iOS platform

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

9.2.1.1 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).
- 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".

9.2.1.2 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"]];
   }
```

9.2.1.3 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). Vipps mobile application will use below code to launch merchant application.

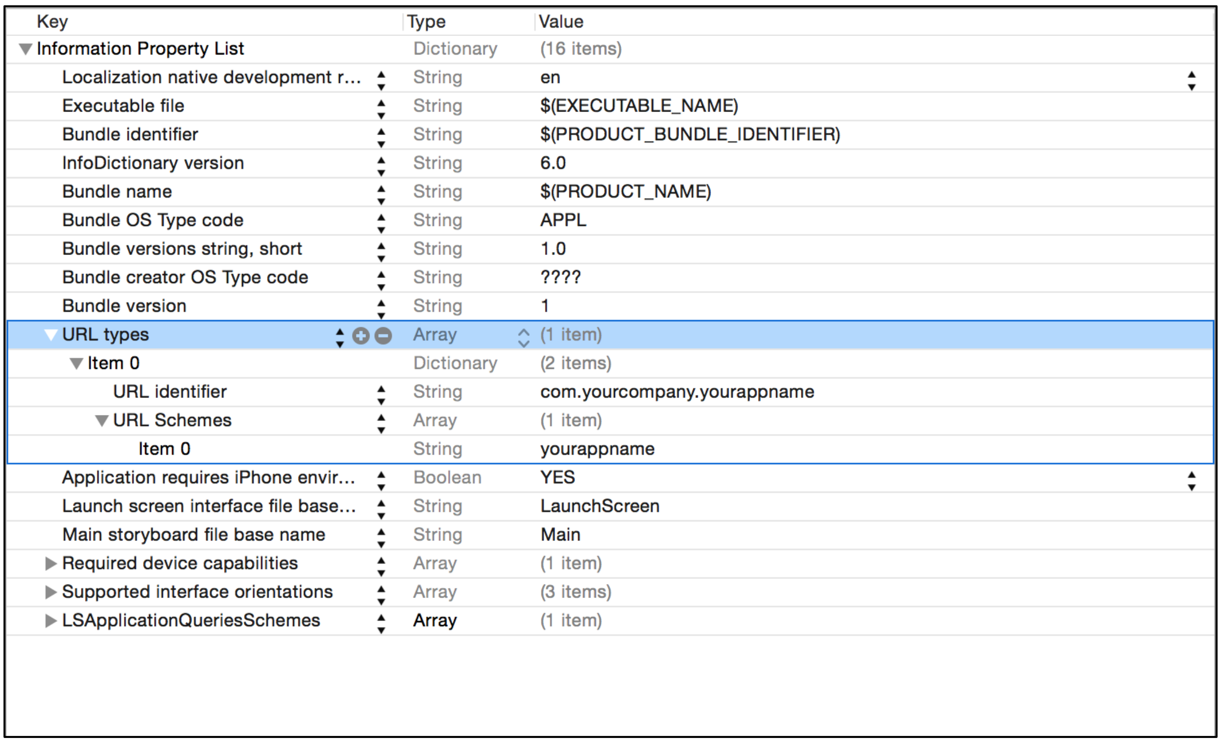
```
NSString *fallbackURL = self.fallbackURL; //fallback url will be the url which has been
provided in Vipps API.
NSURLComponents *urlComponents = [NSURLComponents
componentsWithString:fallbackURL];
NSMutableArray <NSURLQueryItem *>*queryItems = [[NSMutableArray
alloc]initWithArray:urlComponents.queryItems];
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”

9.2.1.3.1 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.



Key	Type	Value
Information Property List	Dictionary	(16 items)
Localization native development r…	String	en
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
URL types	Array	(1 item)
Item 0	Dictionary	(2 items)
URL identifier	String	com.yourcompany.yourappname
URL Schemes	Array	(1 item)
Item 0	String	yourappname
Application requires iPhone envir…	Boolean	YES
Launch screen interface file base…	String	LaunchScreen
Main storyboard file base name	String	Main
Required device capabilities	Array	(1 item)
Supported interface orientations	Array	(3 items)
LSApplicationQueriesSchemes	Array	(1 item)

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.
}
```

9.2.2 Appswitch with Android platform

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

9.2.2.1 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”.

9.2.2.2 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

9.2.2.2.1 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);
}
```

9.2.2.2.2 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);
}
```

9.2.2.3 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:

9.2.2.3.1 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();
               }
            }
        }
    }
  }
}
```

9.2.2.3.2 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();
      }
   }
}
```

9.3 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

10. API definitions

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

10.1 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.
Content-Type	application/json	No	Type of the body
X-TimeStamp	Time stamp when the request called	Yes	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

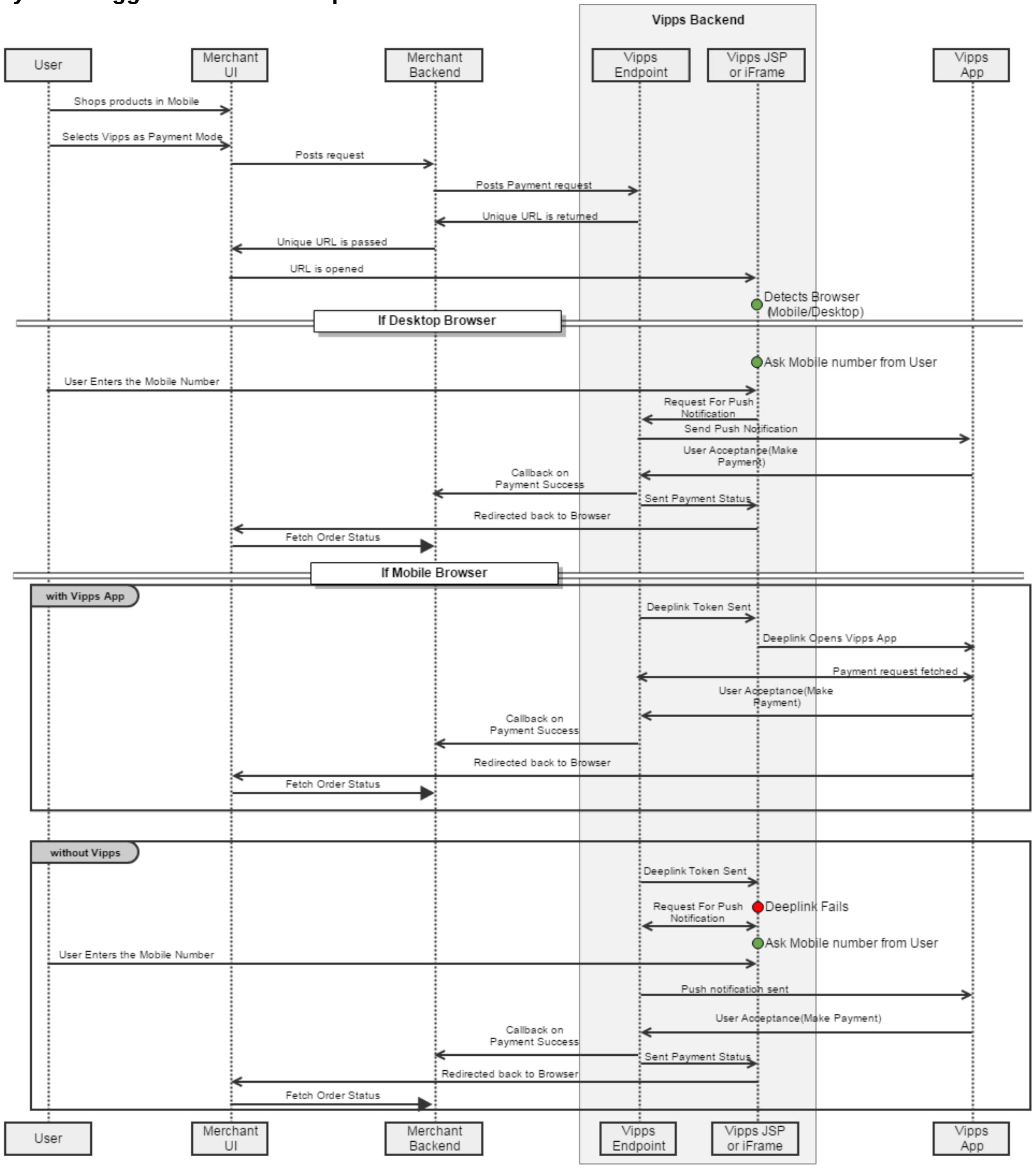
10.2 Initiate Payment

Initiate payment is used to create a payment order in Vipps. In order to identify sales channel payments are coming from merchantSerialNumber is used to distinguish between them. Merchant provided orderId must be unique per sales channel. Please note that single payment is uniquely identified by composition of merchantSerialNumber and orderId.

Initiate call will have parameter *paymentType* which will identify regular ecommerce payment and express checkout payment flow.

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

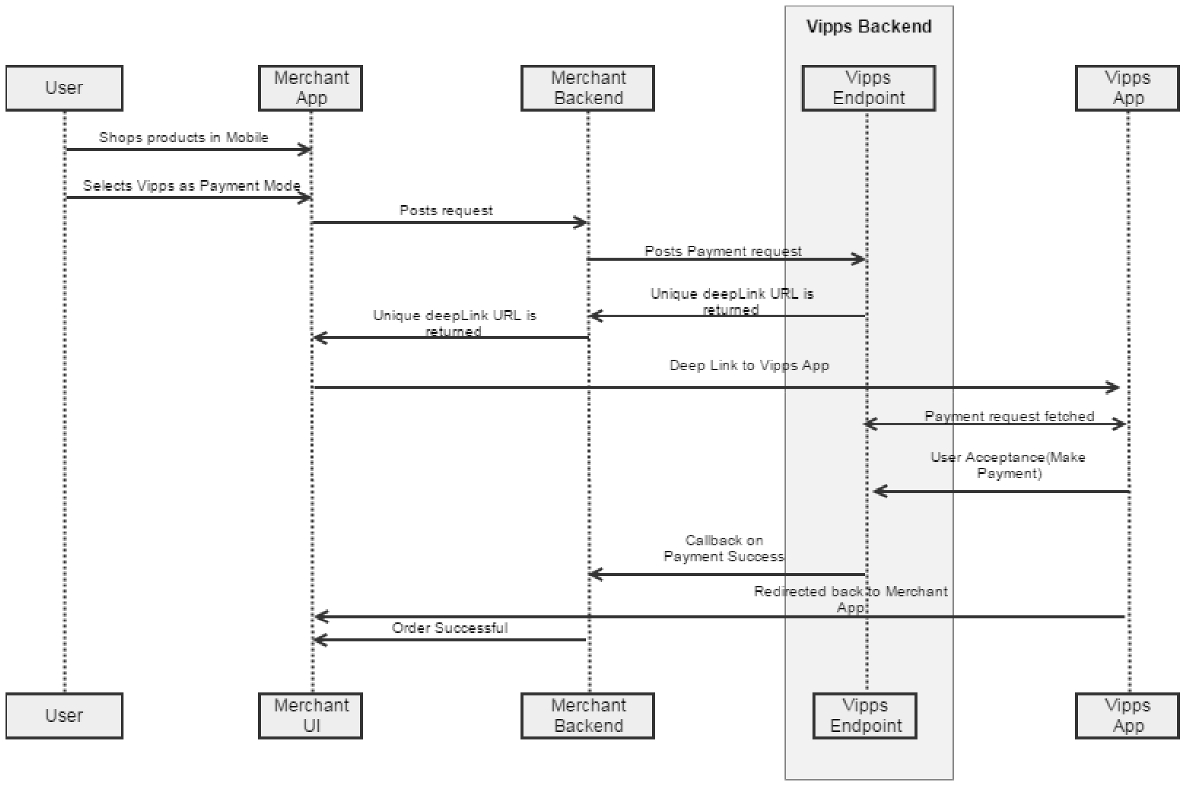
1. Landing page will check if Vipps app is available in same mobile device.
2. If Vipps app is not available then landing page will ask for user's mobile number. After user enters mobile number, Vipps will send push notification to corresponding Vipps profile, if exists. Landing page is not closed in 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.

10.2.1 URL for Initiate Payment

/v2/payments

10.2.2 Method

POST

10.2.3 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://www.domain.no/fallback?sessionId:1234566”,
    “isApp” : “true/false”  , Boolean DEFAULT FALSE
    "merchantSerialNumber": "123456",
    “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"
  }
}
```

10.2.4 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	Yes	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	Yes	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	-	Yes	This will parameter will identify difference between ecomm payment and ecomm express payment.

10.2.5 Success Response

Http Status Code	Content
200	ok

10.2.6 Response Body

```
{
  "orderId": "219930212",
  "url":https://vipps.no/payments/paymentgateway.jsp?token=fasdfqw04rtasdkfam.asdfqw30rsdfasd802354.asdfajsjq3
or Vipps://app?token= fasdfqw04rtasdkfam.asdfqw30rsdfasd802354.asdfajsjq340sfa"

  }
}
```

10.2.7 Description

Id	Type	Size	Optional	Description
orderId	String	30	No	Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters
url	String	-	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

10.2.8 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

10.3 Cancel Payment

Cancel payment call allows merchant to cancel a reserved payment order

10.3.1 URL

/v2/payments/{orderId}/cancel

10.3.2 Method

PUT

10.3.3 URLParams

orderId =[Integer | String] REQUIRED

10.3.4 Request Body

```
{
  "merchantInfo": {
    "merchantSerialNumber": "123456"
  },
  "transaction": {
    "transactionText":"transaction text"
  }
}
```

10.3.5 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

10.3.6 Success Response

Http Status Code	Content
200	ok

10.3.7 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
  }
}
```

10.3.8 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

10.3.9 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

10.4 Capture Payment

Capture payment 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 as long as 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.

10.4.1 URL

/v2/payments/{orderId}/capture

10.4.2 Method

POST

10.4.3 URLParams

orderId =[Integer | String] REQUIRED

10.4.4 Request Body

```
{
  "merchantInfo": {
    "merchantSerialNumber": "123456"
  },
  "transaction": {
    "amount": 1200,
    "transactionText":"transaction text"
  }
}
```

10.4.5 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

10.4.6 Success Response

Http Status Code	Content
200	ok

10.4.7 Response Body

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

10.4.8 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

10.4.9 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

10.5 Refund Payment

Refund payment allows merchant to do a refund of an already captured payment order. There is an option to do a partial refund of the captured amount by giving an amount which is lower than the captured amount. Refunded amount cannot be larger than captured.

10.5.1 URL

/v2/payments/{orderId}/refund

10.5.2 Method

POST

10.5.3 URLParams

orderId =[Integer | String] REQUIRED

10.5.4 Request Body

```
{
   "merchantInfo": {
      "merchantSerialNumber": "123456"
   },
   "transaction": {
      "amount":1200,
      "transactionText":"Transaction text"
   }
}
```

10.5.5 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

10.5.6 Success Response

Http Status Code	Content
200	ok

10.5.7 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"
  }
}
```

10.5.8 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

10.5.9 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

10.6 Get Payment Details

Get Payment Details allows merchant to get the details of a payment order. Service call returns detailed transaction history of given payment where events are sorted by the time.

10.6.1 URL

v2/payments/{orderId}/details

10.6.2 Method

GET

10.6.3 URLParams

orderId =[Integer | String] REQUIRED

10.6.4 Success Response

Http Status Code	Content
200	ok

10.6.5 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
  }
}
```

10.6.6 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

10.6.7 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

10.6.8 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.

10.6.9 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

10.7 Get Order Status

Get Order Status allows merchant to get the status of a payment order.

10.7.1 URL

v2/payments/{orderId}/status

10.7.2 Method

GET

10.7.3 URLParams

orderId =[Integer | String] REQUIRED

10.7.4 Success Response

Http Status Code	Content
200	ok

10.7.5 Response Body

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

}
```

10.7.6 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

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

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.

10.8.1 Request Header

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

10.8.2 URL

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

10.8.3 Method

POST

10.8.4 URLParams

orderId =[String] REQUIRED

10.8.5 Success Response

Http Status Code	Content
200	Ok

10.8.6 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
}
```

10.8.7 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
  }]

}
```

10.8.8 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

10.8.9 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

10.9 Callback : Transaction Update

Callback allows Vipps to send the payment order 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.

If the communication is broken during payment process for some reason, and Vipps is not able to execute callback, then callback will not be retried.
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.

10.9.1 Request Header

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

10.9.2 URL

[callbackPrefix]/v2/payments/{orderId}

10.9.3 Method

POST

10.9.4 URLParams

orderId =[String] REQUIRED

10.9.5 Success Response

Http Status Code	Content
200	Ok

10.9.6 Request Body

Regular ecommerce 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": ""
 },
}
```

10.9.7 Response Body

NA

10.9.8 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.

10.10 Remove User Consent (for express checkout)

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.

10.10.1 URL

{consetRemovalPrefix}/v2/consents/{userId}

10.10.2 Method

DELETE

10.10.3 URLParams

userId =[String]   REQUIRED

10.10.4 Success Response

Http Status Code	Content
200	ok

10.10.5 Request Body

NA

10.10.6 Response Body

NA

10.10.7 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