Vipps – eCommerce & InApp API

Version: **2.1**
Date produced: **6-November-2017**

Content

Overview

Vipps eCommerce API gives merchant a great control over Vipps payment lifecycle. It gives possibility to initiate payment from different sales channels like a mobile app, web-shop, customer care systems, automated POS etc. It also enables merchant to affect payment flow by utilizing functions like payment reservation, capture, cancellation and refunding.

Target Audience

Target audiences for this document are solution/system architect and developers. Even though document is of quite technical nature, others like a business owners can benefit as well, since functions are described also from high level perspective.

Use case scenarios

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

A. Mobile app payments

Vipps eCommerce API enables payments by Vipps made through merchant mobile application. In order to do this InApp Deeplinking section describes in detail how to call Vipps App from merchant App and initiate payment flow. As described in section, there are two types of payment flows that can be initiated, direct capture one, and one that uses reserve/capture features.

For direct capture payment flow following API service calls should be implemented:

Get Order Status – Server side check that payment is confirmed
Refund Payment – Optional. Used for refund payments

For reserve/capture payment flow following API service calls should be implemented:

Get Order Status – Server side check that payment is confirmed
Cancel Payment – Used for cancel payment reservation
Capture Payment – Used for capture reserved payment
Refund Payment – Optional. Used for refund payments

B. Webshop payments

Vipps eCommerce API enables payments by Vipps made through merchant webshop. For reserve/capture payment flow following API service calls should be implemented:

Feil! Fant ikke referansekilden. – Server side check that payment is confirmed
Cancel Payment – 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

C. Mobile webshop payments

Vipps eCommerce API enables payments by Vipps made through mobile version of merchant webshop. This use case is hybrid one, a combination of previous two. It

applies when merchant have responsive design of the webshop that change layout depending of device capabilities. Merchant have ability to implement both inApp deeplinking style and classic eCommerce style of payment initiation. In order to enable the first one, InApp Deeplinking section describes in detail how to call Vipps App from mobile version of merchant webshop and initiate payment flow. As described in section, there are two types of payment flows that can be initiated, direct capture one, and one that uses reserve/capture features. There are two constrains using this inApp Deeplinking style. First, merchant must be able to detect mobile version of the web browser and dynamically enable deeplinking for mobile phones. The second constraint is that merchant must provide URI for Vipps App to execute fallback.
Making such dynamic integration with Vipps will enhance user experience by using Vipps as payment method on mobile devices.

For direct capture payment flow following API service calls should be implemented:

- Get Order Status – Server side check that payment is confirmed
- Refund Payment – Optional. Used for refund payments

For reserve/capture payment flow following API service calls should be implemented:

- Get Order Status – Server side check that payment is confirmed
- Cancel Payment – Used for cancel payment reservation
- Capture Payment – Used for capture reserved payment
- Refund Payment – Optional. Used for refund payments

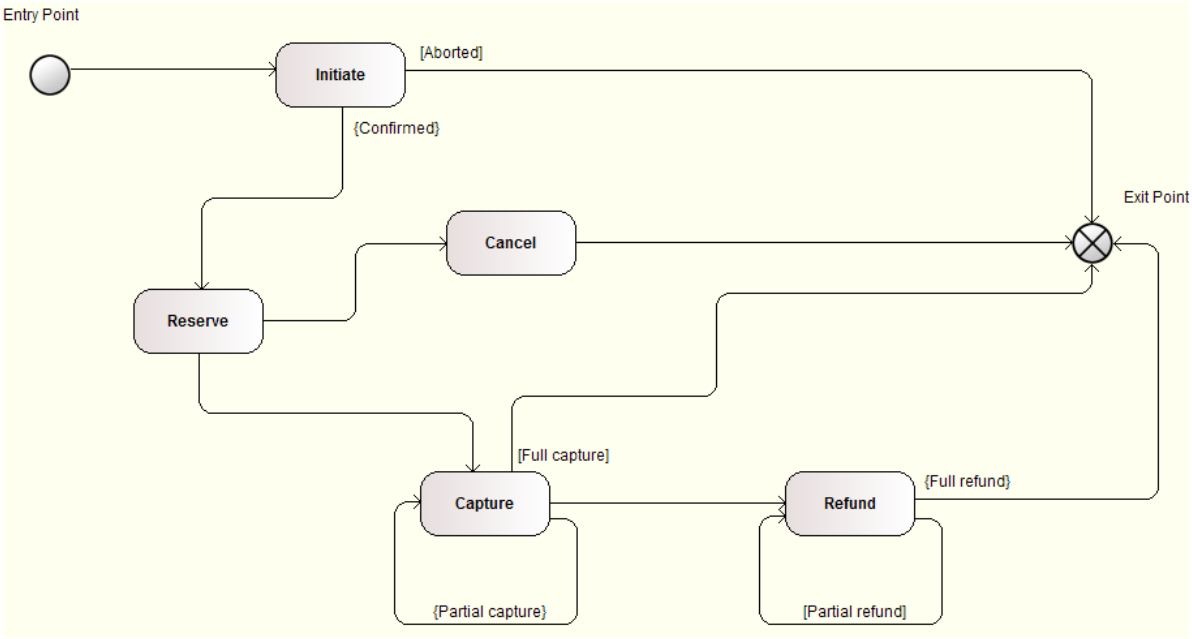
D. Physical stores payments

Vipps eCommerce API enables payments by Vipps made through merchant stores with direct capture. Direct capture is initiated using the Initiate payment request and configurations in Vipps backend for merchantSerialNumber will trigger Direct capture. For direct capture payment flow following API service calls should be implemented:

- **Feil! Fant ikke referansekilden.** – Server side check that payment is confirmed
- Refund Payment – Optional. Used for refund payments
- Get Payment Details – Optional. Used to retrieve transaction history

1. Payment flow

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



1.1 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.

1.2 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.

1.3 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.

1.4 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.

1.5 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.

1.6 Refund

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

1.7 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 are 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
AUTO_REVERSAL	Transaction timed out at PSP, 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

1.8 Get Payment Details

Get Payment Details will provide all the payment transaction details.

Operation	Description
INITIATE	Payment Initiated
REGISTER	When we call PSP for payment
RESERVE	Payment reserved
SALE	Direct Capture
CANCEL	If user doesn't take action on payment request
REJECTED	User rejected the payment
CAPTURE	Card is charged
REFUND	Money refunded to user
VOID	Reservation cancelled
AUTO_REVERSAL	Transaction timed out at PSP, we will refund the transaction and then it will be autoreversal

2. Idempotency

Initiate, capture and refund calls are idempotent in Vipps eCommerce and 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 beretried with the same idempotent key. Idempotent key is generated by merchant.

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

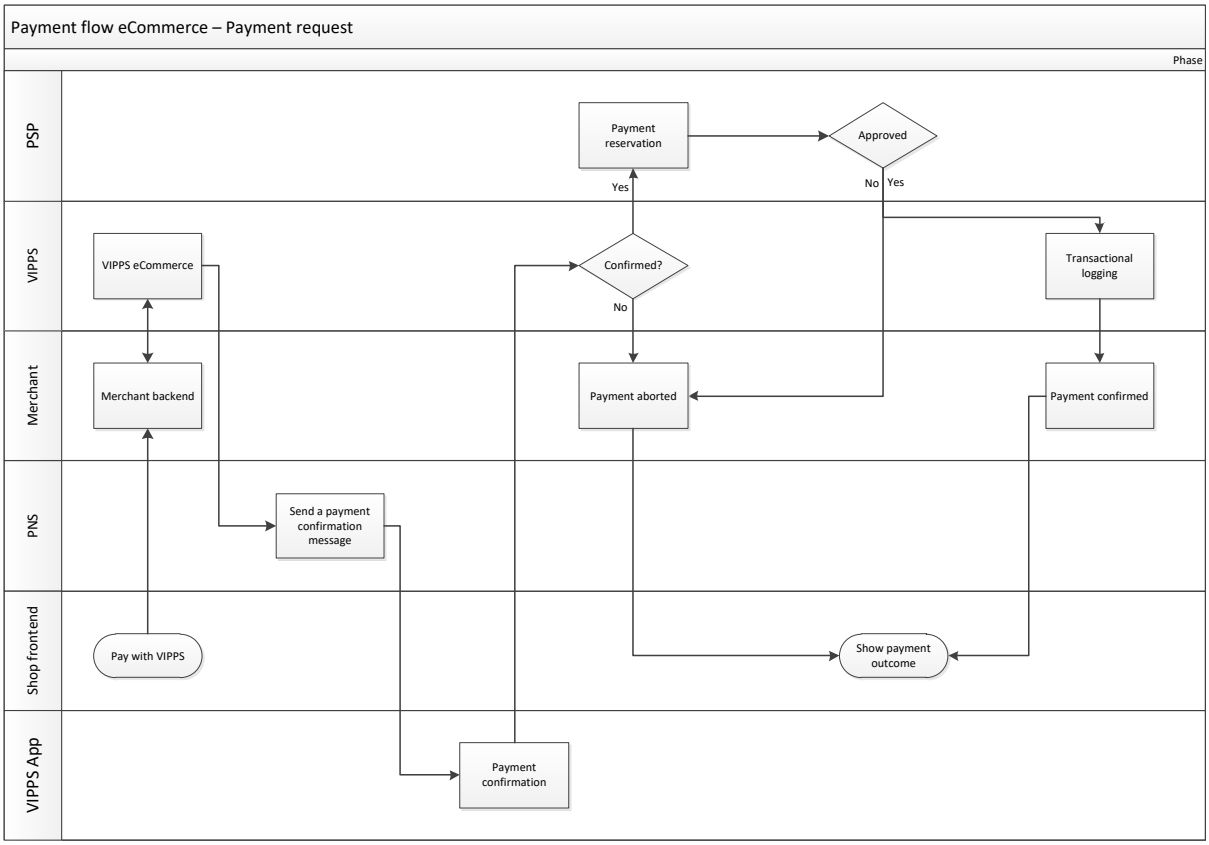
3. Exception handling

3.1 Introduction

Every system, especially those that includes complex integrations and/or participation of many users, is prone to unexpected conditions. Since we can't affect some of them, like communication interruption, we can decide how we cope with them in order to minimize impact they impose. Some of measures are described briefly there they naturally belongs to (i.e. payment request) and in this chapter we are explaining how the merchant should react when exception occurs.

3.2 Exception scenarios

The most critical action in payment flow is when **Feil! Fant ikke referansekilden.** service call is evoked. Flow diagram below shows that 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.

3.2.1 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 10 seconds socket read timeout while communicating with Vipps.

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

3.2.2 Callback aborted/interrupted

Please note that callback can be executed at any time within timeframe of 4 minutes after payment request is sent. With other words, if the merchant doesn't receive any confirmation on payment request call within callback timeframe, request should be treated as aborted and Cancel Payment service call should be executed in order to ensure transaction consistency.

3.2.3 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.

4. Error responses

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

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

field	example	description
errorCode	"01"	error code which uniquely identifies an error scenario, see 4.4
errorGroup	"Authentication"	Error category, see 4.3
errorMessage	The provided credentials doesn't match	Detailed description of the error if errorMessage doesn't provide necessary information, Merchant need to log this description to get the details of the error message

4.1 Error Groups

Error Groups	Description
Authentication	Authentication Failure because of wrong credentials 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

4.2 Error codes

Error Group	Error Code	Error Message
Authentication	01	Provided credentials doesn't match
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

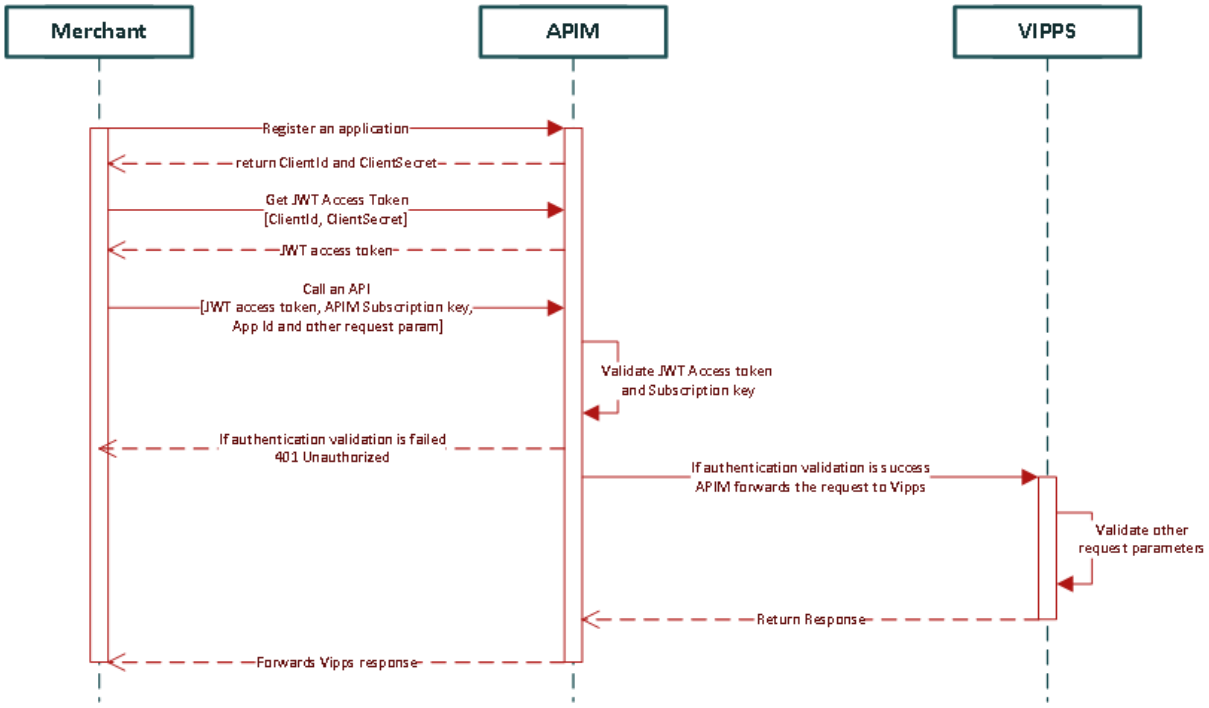
4.2.1 List of error codes for inApp deeplinking

Error Codes	Description
100	Transaction Success
201	Invalid Input Parameter
202	Transaction Cancelled By User
301	Invalid Merchant details or Merchant doesn’t have active account
302	User Doesn’t has Vipps Profile
303	Login Failed (Login Max attempt reached)
304	VIPPS doesn’t support this action, Please update Vipps
401	Transaction Timed Out
402	User’s Transaction Limit Reached (Daily/Weekly/Monthly/Yearly)
403	Transaction Failed
451	The user was selected for fraud validation
999	Failed

5. Merchant enrollement

Merchant enrollment process consists of two steps. The first step is the mercantile steps of signing the Vipps agreement after which Vipps completes the registration. After registration merchant receives username and password to login into Vipps Developer Portal. In the portal merchant can complete the second step, which is to register an application to generate merchant credentials.

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 below steps:

- Merchant logs into Vipps Developer portal and register an application that will consume Vipps apis. On successful registration, it will receive application credentials (<ClientId> and <ClientSecret>).
- Merchant application use the <ClientId> and <ClientSecret> to get a JWT access token. JWT access token is a base 64 encoded string value that needs to be used as a bearer token in request header.
- Merchant application use this JWT access token, a unique subscription key and the application id <ClientId> to authenticate all subsequent calls to Vipps API. See tabell below for technical specification.
    - If request is not authenticated 401 Unauthorized is returned in the response.
- When authenticated Vipps process the request and produce corresponding response to merchant.

Authentication headers when calling Vipps API

Header Name	Header Value	Description
Authorization	"Bearer <jwt access token>"	type: Authorization token value: JWT access token is obtained by calling the access token service.
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

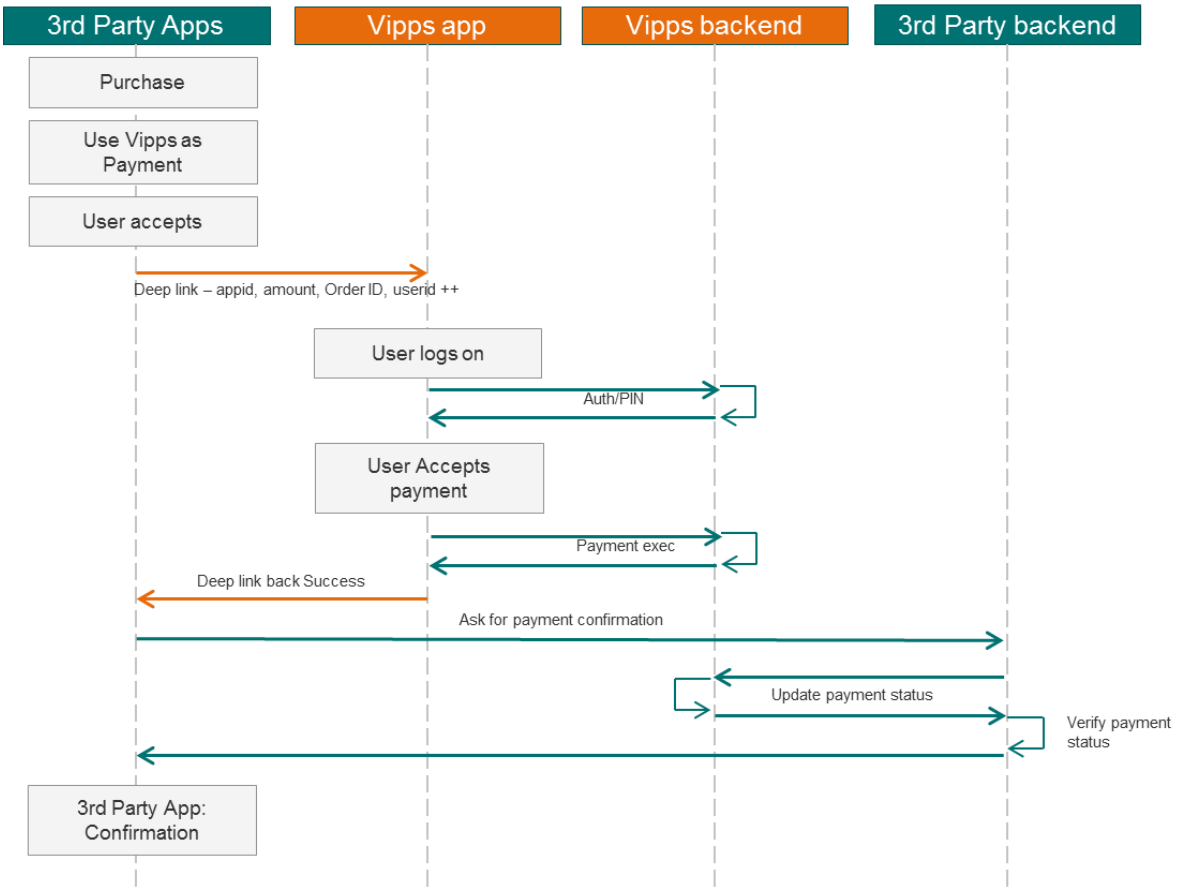
6. InApp Deeplinking

6.1 Introduction

Deeplinking provides application interoperability between a native app or web view, and the native Vipps application. Implementation of *deeplinks* varies based on platforms.

Vipps supports *Deeplinking* from any authentic application for receiving payments for Products / Services / Charity. Payment requested through *Deeplinking* will be, through standard set of parameters passed from registered 3rd party applications and Vipps, processed as any other eCommerce payment.

6.2 Deeplinking flow



6.3 Status codes

Webservice Get Order Status returns the last order status where user interaction was mandatory. That means only three statuses are available; CANCEL when user deliberately refused to confirm a payment request, RESERVE when user confirmed a payment request in reserve/capture (action = inAppReserve) payment flow and SALE when user confirmed a payment request in direct capture (action = inAppPayment) payment flow. When Get Order Status returns CANCEL status, merchant should not consider payment as successful.

6.4 iOS implementation

6.4.1 Vipps App detection

For *deeplinking* on iOS, first check to see if the Vipps app is installed on the user's device. If the Vipps app is installed, launch the VPPSs app; if the Vipps app is not installed, launch Vipps's app store web link (https://itunes.apple.com/no/app/Vipps-by-dnb/id984380185)

To determine if the Vipps app is installed on iOS, use the following call:

```
NSString url = @"vipps://action=inAppPayment&…";
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"]];
}
```

6.4.2 Launching Vipps App

To launch the Vipps app, use the following call (at a minimum):

```
vipps://?action=inAppPayment&appID=<Your Registered App Id>&amount=12000&merchantSerialNumber=<your serial number from Vipps backend system>&fallbackURL=<URL to which Vipps should respond after transaction>
```

You can add additional query parameters as listed below. Please refer to the examples below as needed.

6.4.3 Request query parameters

Passing parameters can make the Vipps experience even more seamless for your users. Parameters used in linking must be URL-encoded with %20 for spaces.

Name	Type	Description
action	string (mandatory)	for payment it should be "inAppPayment" and for reservation "inAppReserve".
appID	string (mandatory)	App registration id will be created for Merchants' each app registered in Vipps Systems and will be shared once the enrolment process is done
amount	integer (mandatory)	Amount for which the payment should be initiated. The amount should in the lowest denomination i.e., Øre Vipps only support NOK currency. Minimum value is 100 and maximum value is 9999999. Vipps will not round of the provided number. If the amount is not in the mentioned format, Vipps

		will return the FAIL state of transaction mentioning incorrect number format.
orderID	string (mandatory)	Order id from 3rd party app for which user is initiating a payment. Maximum length for this field is 30 chars. This order ID will be stored in the Vipps systems against the transactions and shall be displayed the receipt in the Vipps App for the user. The order ID will also be presented for the respective transaction in the Sales and Settlement report.
message	string (optional)	URL encoded string. Payment's reference message. Maximum length for this field is 200 chars. The message field will be displayed as part of the receipt.
merchantSerialNumber	string (mandatory)	Merchant Sales Unit's serial number will be a unique number in Vipps systems. The Serial number registered in Vipps Systems will be shared once the enrollment process is done.
data	string (optional)	URL encoded string. If this exists, then the contents will go as a query string parameter with fallbackURL. Calling app can use this field to identify their session or booking for which the payment has been done. Maximum length for this field is 128 chars.
fallbackURL	string (mandatory)	URL encoded string. URL to be invoked once the payment process is complete. This is to respond back to calling app.

Example

```
vipps://?action=inAppPayment&appID=fa01s-21res-qq21p-
mq1p6&amount=12000&merchantSerialNumber=ed9320pre&fallbackURL=app://?action=confirmed&
message=Test%20Message&data=<any URL encoded string>
```

6.4.4 Launching back source application

After processing the transaction, Vipps application will open following url. Vipps application will not validate the content of fallbackURL. Vipps **application will** URLDecode the fallbackURL parameter. Then the query string parameters will be appended to the decoded fallbackURL. After that **Vipps** will open the url.
Sample code in Vipps for opening the URL:

```
if ([[UIApplication sharedApplication] canOpenURL:[NSURL URLWithString:@"
fallbackURL&status=SUCCESS&…"]]) {
    // Navigating back to source application
        [[UIApplication sharedApplication] openURL:[NSURL URLWithString: @"
fallbackURL&status=SUCCESS&transactionId=5001236541&…"]];
}
```

Sample callback url:

```
fallbackURL&status=SUCCESS&transactionId=5001236541&data=<whatever the data has been
sent in input
```

6.4.5 Response query parameters

Name	Type	Description
status	string	This field will have various codes by which calling app can

	(mandatory)	handle error cases. The list of error codes are detailed in section 2.2.2
errorMessage	string (optional)	If the transaction status is "999", Vipps will provide error message in English mentioning the reason behind the failure.
transactionId	string (optional)	If transaction is success then only this parameter will be returned. This will be a number with maximum of 30 digits which can uniquely identify the transaction.
data	string (optional)	If the parameter "data" has been passed as an input parameter then same value will be returned in this field.
orderID	string (mandatory)	Vipps will reply back with same orderID that was passed in deeplink input query parameter.

6.4.6 Registering 3rd Party app with URL Schema 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

6.5 Android OS implementation

6.5.1 Vipps App detection

For deep linking on Android, first check to see if the Vipps app is installed on the user's device. If the Vipps app is installed, check the version of Vipps app. If version of Vipps is less than "X" then deep linking feature is not enabled in the app. Calling app needs to handle this case by showing appropriate information to user. Note that, in below code, X is assumed to be 1.4.0. This may change during design based on business needs. Version number string will be confirmed in the final release of this document.

If the Vipps app is not installed, launch Vipps's play store page (https://play.google.com/store/apps/details?id=no.dnb.Vipps). Here's an example:

```
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 =
      "vipps://?action=inAppPayment&appID=<Your Registered App
Id>&amount=120.00&merchantSerialNumber=<your serial number from Vipps backend
system>&fallbackURL=<URL to which Vipps should respond after transaction>";
         Intent intent = new Intent(Intent.ACTION_VIEW);
         intent.setData(Uri.parse(uri));
         startActivity(intent);   //Approach 1 explain below
                  OR
         startActivityForResult(intent,requestCode);   //Approach 2 explain below
```

```
  } 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);
}
private Integer versionCompare(String str1, String str2) {
    String[] vals1 = str1.split("\\.");
    String[] vals2 = str2.split("\\.");
    int i = 0;
    // set index to first non-equal ordinal or length of shortest version string
    while (i < vals1.length && i < vals2.length && vals1[i].equals(vals2[i])) {
      i++;
    }
    // compare first non-equal ordinal number
    if (i < vals1.length && i < vals2.length) {
        int diff = Integer.valueOf(vals1[i]).compareTo(Integer.valueOf(vals2[i]));
        return Integer.signum(diff);
    }
    // the strings are equal or one string is a substring of the other
    // e.g. "1.2.3" = "1.2.3" or "1.2.3" < "1.2.3.4"
    Else {
        return Integer.signum(vals1.length - vals2.length);
    }
}
```

Android supports the same URL and query parameters as iOS, so please refer to the input query parameters and response query parameters section for iOS for more information and examples.
3rd party application can integrate Vipps by following 2 approaches

- Custom URL Schema
- Android's intent system

6.5.2 Custom URL Schema

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

6.5.2.1 a. Set filter in Manifest file:

To receive call back from Vipps application to activity first need to set filter to that activity. In below example MainActivity is the receiving activity. Vipps application sends response to this activity. For this activity can set custom URL schema inside 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 input query

fallbackURL=<URL to which Vipps should respond after transaction> .

For Example:

```
vipps://?action=inAppPayment&appID=<Your Registered App
Id>&amount=120.00&merchantSerialNumber=<your serial number from Vipps backend
system>&fallbackURL=sampleApps://&message=Test%20Message&data=<any URL encoded string>
```

and in Manifest file add <data android:scheme="**sampleApps** " /> in filter of receive activity.

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

6.5.2.2 Start Vipps application

Start Vipps application by setting URL to Intent and call startActivity with specified Intent.

For Example:

```
String uri =
      "vipps://?action=inAppPayment&appID=<Your Registered App
Id>&amount=120.00&merchantSerialNumber=<your serial number from Vipps backend
system>&fallbackURL=<URL to which Vipps should respond after transaction>";

String encodedURL = URLEncoder.encode(uri,"UTF-8");

Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse(encodedURL));
```

```
startActivity(intent);
```

6.5.2.3 Handle Query String in Java code:

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

For Example:

```
@Override
protected void onNewIntent(Intent intent) {
       super.onNewIntent(intent);

       String url = null;
       if (data != null && data.getExtras() != null) {
            try {
                Bundle mBuddle = data.getExtras();
                   if (mBuddle.get("data") != null)
                     url = URLDecoder.decode(mBuddle.get("data").toString(), "UTF-8");

            } catch (UnsupportedEncodingException e) {
                        e.printStackTrace();
            }
            return url;
}
}
```

intent.getData() will contain query string.

6.5.3 Android's Intent system

Vipps application support intent base deep link approach like other social apps. If 3rd application does not want use custom URL scheme approach and go with Intent base standard approach then they can integrate Vipps by this approach. In this approach there is no need to set filter for call back receiving activity, just need to register activity in manifest file. Start Vipps application by calling startActivityForResult and receive result in onActivityResult by overriding this method.

6.5.3.1 Register the Activity in Manifest file:

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>
```

6.5.3.2 Start Vipps application

Start Vipps application by setting URL to Intent and call startActivityForResult with specified Intent.

For Example:

```
String uri =
      "vipps://?action=inAppPayment&appID=<Your Registered App
Id>&amount=120.00&merchantSerialNumber=<your serial number from Vipps backend
system>&fallbackURL=INTENT";

String encodedURL = URLEncoder.encode(uri,"UTF-8");

Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse(encodedURL));
startActivityForResult(intent);
```

Note: You need to specify INTENT in fallbackURL. If you not send this in fallbackURL then your application will not receive any response from Vipps application.

However, if you prefer to start Vipps application directly and bypass the system picker, you can do so by using setPackage in your intent.

```
intent.setPackage("no.dnb.Vipps");
```

6.5.3.3 Handle Query String in Java code:

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

For Example:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

	if(resultCode == RESULT_OK) {
		String url = null;
		if (intent != null && intent.getData() != null) {
			try{
				url = URLDecoder.decode(intent.getData().toString(),"UTF-
8");
				//TODO Handle result

			}catch(UnsupportedEncodingException e) {
				e.printStackTrace();
			}

		}

	}
}
```

7. API definitions

7.1 Access Token

7.1.1 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.

7.1.2 URL

https://portal.apivipps.no/accessToken/get

7.1.3 Method

POST

7.1.4 Request Headers

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

7.1.5 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

7.1.6 Success Response

Http Status Code	Content
200	OK

7.1.7 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":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6InowMzl6ZHNGdWl6cEJmQlZLMVRuMj
VRSFlPMCIsImtpZCI6InowMzl6ZHNGdWl6cEJmQlZLMVRuMjVRSFlPMCJ9.eyJhdWQiOiIwMDAw
MDAwMi0wMDAwLTAwMDAtYzAwMC0wMDAwMDAwMDAwMDAiLCJpc3MiOiJodHRwczovL3N0cy53aW5
kb3dzLm5ldC9lNTExNjUyNi01MWRjLTRjMTQtYjA4Ni1hNWNiNDcxNmJjNGIvIiwiaWF0IjoxND
k1MTg0NTc0LCJuYmYiOjE0OTUxODQ1NzQsImV4cCI6MTQ5NTI3MTI3MywiYWlvIjoiWTJaZ1lGQ
VdFcmdncnJWbS9wY3ZENDJ6NzI1NUJnQT0iLCJhcHBpZCI6ImQ5YjZjOTlkLTI0NDItNGE1ZC04
NGEyLWM1M2E4MDdmZTBjNCIsImFwcGlkYWNyIjoiMSIsImlkcCI6Imh0dHBzOi8vc3RzLndpbmR
vd3MubmV0L2U1MTE2NTI2LTUxZGMtNGMxNC1iMDg2LWE1Y2I0NzE2YmM0Yi8iLCJ0aWQiOiJlNT
ExNjUyNi01MWRjLTRjMTQtYjA4Ni1hNWNiNDcxNmJjNGIiLCJ1dGkiOiJuYTJBczl2SWlVMm12U
TRzVUprRkFBIiwidmVyIjoiMS4wIn0.QaopYuvN8jsh82uepLcF-
uLqEhdFsRNl6_KrjAva537HHMa2x6w2pL2v96k40QBjD8A_GGHZ-E2VC3QSY-
WsPdHUI5Kb4zEQzJ4-_CnMRo3bXavz3Sdo2-
1amFKsOY8AFODpqJR0MYqPK_Kr6sSIWL3M_L3wu0rG976HIXllsRLvWBSwDeMgBAUvwWrXCmnVf
znOleSxsPbAxZshn3xjuYeJWEAR7ZpJQhjuGpiu0rP7lERTxX_rCnW1cr3m7RfE16z8e5VQva9A
Ot4OG5NuIrLJqmhb3KHBa3GusK6KLf-pVjF6fnS5r0ZQ5foP-VqOCiK9CUUATjHEOflgy1ubvA"
}
```


7.1.8 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

7.1.9 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

7.1.10 Error Response Body

7.1.10.1 400 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"
}
```


7.1.10.2 401 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"
}
```


7.2 Request Headers

Header Name	Header Value	Optional	Description
Authorization	Bearer <<value>>	No	type: Authorization token value: JWT string received from access token service
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-Source-Address	Either source ip address or device id and terminal id for mobile application	No	Identifying the request source
X-JWT	Base 64 encoded Json web token but to be used in future	Yes	Json Web token Base 64 encoded
X-Request-Id	To identify the idempotent request	No/Yes	For Making request to be idempotent this ID is must so that the system will not do any side effects. 1. Mandatory for Initiate,Capture, Refund payment 2. Size should be max 30 alphanumeric 3. Optional for Cancel Payment, Get Payment Details and Get

			Order Status. 4. If user wants to re-try any failed capture or refund transaction then they should provide same X-request-id, else system will create a new entry for partial capture or partial refund.
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

7.3 Initiate Payment

7.3.1 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 *merchantSerialNumber* and *orderId*.

Merchant provided *orderId* must be unique per sales channel which leads to that there will be no new payment flow initiated for repetitive use of the same *orderId* in initiate payment service call.

After the customer has been 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.

Vipps answers immediately to service call and rest of processing is asynchronous. After processing is done, Vipps will execute callback to the registered URL with the status of the request. The callback call will be made via HTTPS, without any credentials. Callback is sent once. Please note that callback can be executed at any time within timeframe of 5 minutes after payment request is sent. With other words, if the merchant doesn't receive any confirmation on payment request call within callback timeframe, request should be treated as aborted and Cancel Payment service call should be executed in order to ensure transaction consistency.

7.3.2 URL

```
/v1/payments
```

7.3.3 Method

```
POST
```

7.3.4 Request Object

```
{
    "merchantInfo": {
```

```
        ”merchantSerialNumber”: “123456”,,
        "callBack": "https://www.demo.no/api/callback"
    },
    "customerInfo": {
        "mobileNumber": "90090900"
    },
    "transaction": {
        "orderId": "219930212",
        "refOrderId": "119930211",
        "amount": 1200,
        "transactionText": "Transaction text",
        "timeStamp":"2014-06-24T08:34:25-07:00"
    }
}
```

7.3.5 Description

Id	Type	Size	Optional	Description
merchantSerialNumber	String	6	No	Identifies a merchant sales channel i.e. website, mobile app etc.
callBack	String	255	No	Vipps will use this Call back URL to provide the status of initiated transaction. Length of the string is 255 characters
mobileNumber	String	-	No	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

7.3.6 Success Response

Http Status Code	Content
202	Accepted

7.3.7 Response Body

```
{
    "orderId": "219930212",
    ”merchantSerialNumber”: “123456”,,
    "transactionInfo": {
        "amount" : 1200,
        "status": "Initiate",
        "transactionId": "1001234566",
        "timeStamp": "2014-06-24T08:34:25-07:00",
        "message":"Please use Vipps app to process the payment"
    }
}
```

7.3.8 Description

Id	Type	Size	Optional	Description
orderId	String	30	No	Id which uniquely identifies a payment. Maximum length is 30 alphanumeric characters
merchantSerialNumber	String	6	No	Identifies a merchant sales channel i.e. website, mobile app etc.
amount	Integer	-	No	Ordered amount in øre. 32 Bit Integer (2147483647)
status	String	-	No	Status of the ordered transaction
transactionId	String	30	No	Vipps transaction id
timeStamp	String	-	No	Timestamp in ISO-8601 representing when Vipps did the requested operation
message	String	100	Yes	Message to the user which needs to be displayed in eCommerce site, mostly asking him to look in the Vipps to authenticate the payment

7.3.9 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
403	Forbidden	The request has been understood but failed because of some reason
5xx	Internal server error	Internal server error

7.3.10 Callback URL

```
POST {Merchant Provided URL}
Content-Type: application/json


Example :
POST https://www.demo.no/api/callback
Content-Type: application/json
```

7.3.11 Callback Request Object

```
Example 1:
{
"endPoint":null,
"orderId":"VT1472048925",
"siteId":null,
"transactionInfo":{
"amount":100,
"timeStamp":"2014-06-24T08:34:25-07:00",
"status":"RESERVED",
"transactionId":"101344195"},
"errorInfo":null
}
```

```
Example 2(With Error Message):
{
"endPoint":null,
"orderId":"1475572136239",
"siteId":null,
```

```
"transactionInfo":{
"amount":141,
"timeStamp":"2014-06-24T08:34:25-07:00",
"status":"RESERVE_FAILED",
"transactionId":"100027827"},
"errorInfo":{
"errorCode":"41",
"errorGroup":"Payment",
"errorMessage":"User don't have a valid card"}
}
```


7.3.12 Callback Request Object Description

Id	Type	Size	Optional	Description
endPoint	String	-	Yes	endPoint always have no value. It can be ignored.
orderId	String	30	No	Id which uniquely identifies a payment
siteId	String	-	Yes	siteId can be ignored
amount	Integer	-	No	Ordered amount in øre. 32 Bit Integer (2147483647)
status	String	-	No	Status of the ordered transaction (RESERVED | CANCELLED | RESERVE_FAILED | NO ACTION | SALE)
transactionId	String	30	No	Vipps transaciton id
timeStamp	String	-	No	Timestamp in ISO-8601 representing when Vipps reserved Transaction.
message	String	100	Yes	Message to the user which needs to be displayed in eCommerce site, mostly asking him to look in the Vipps to authenticate the payment
errorCode	String	3	Yes	Error code if reservation failed
errorGroup	String	100	Yes	Error Group if reservation failed
errorMessage	String	100	Yes	Error Message if reservation failed

7.4 Cancel Payment

7.4.1 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.

7.4.2 URL

```
/v1/payments/{orderId}/cancel
```

7.4.3 Method

```
PUT
```

7.4.4 URLParams

```
orderId =[Integer | String]    REQUIRED
```

7.4.5 Request Object

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

7.4.6 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

7.4.7 Success Response

Http Status Code	Content
200	ok

7.4.8 Response Body

```
{
    "orderId": "219930212",
    "transactionInfo": {
        "amount": 1200,
        "timeStamp": "2014-06-24T08:34:25-07:00",
        "transactionText": "Refrence text",
```

```
        "status": "Cancelled",
        "transactionId": "100025255"
    },
    "transactionSummary": {
        "capturedAmount":0,
        "remainingAmoutToCapture":0,
        "refundedAmount":0,
        "remainingAmountToRefund":0
    }
}
```


7.4.9 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
remainingAmoutToCapture	Integer	No	Total remaining amount to capture
refundedAmount	Integer	No	Total refunded amount of the order
remainingAmountToRefund	Integer	No	Total remaining amount to refund

7.4.10 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

7.5 Capture Payment

7.5.1 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.

7.5.2 URL

```
/v1/payments/{orderId}/capture
```

7.5.3 Method

```
POST
```

7.5.4 URLParams

```
orderId =[Integer | String]    REQUIRED
```

7.5.5 Request Object

```
{
    "merchantInfo": {
        "merchantSerialNumber": "123456"
    },
    "transaction": {
        "amount": 100,
        "transactionText":"transaction text "
    }
}
```

7.5.6 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

7.5.7 Success Response

Http Status Code	Content
200	ok

7.5.8 Response Body

```
{
```

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


7.5.9 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 transaciton id
capturedAmount	Integer	No	Total amount captured
remainingAmoutToCapture	Integer	No	Total remaining amount to capture
refundedAmount	Integer	No	Total refunded amount of the order
remainingAmountToRefund	Integer	No	Total remaining amount to refund

7.5.10 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

7.6 Refund Payment

7.6.1 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.

7.6.2 URL

```
/v1/payments/{orderId}/refund/
```

7.6.3 Method

```
POST
```

7.6.4 URLParams

```
orderId =[Integer | String]   REQUIRED
```

7.6.5 Request Object

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

7.6.6 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

7.6.7 Success Response

Http Status Code	Content
200	ok

7.6.8 Response Body

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

7.6.9 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 transaciton id
capturedAmount	Integer	No	Total amount captured
remainingAmoutToCapture	Integer	No	Total remaining amount to capture
refundedAmount	Integer	No	Total refunded amount of the order
remainingAmountToRefund	Integer	No	Total remaining amount to refund

7.6.10 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

7.7 Get Payment Details

7.7.1 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.

7.7.2 URL

```
v1/payments/{orderId}/serialNumber/{merchantSerialNumber}/details
```

7.7.3 Method

```
GET
```

7.7.4 URLParams

```
orderId =[Integer | String]    REQUIRED
merchantSerialNumber =[Integer | String]   REQUIRED
```

7.7.5 Success Response

Http Status Code	Content
200	ok

7.7.6 Response Object

```
{
   "orderId": "219930212",
    "transactionSummary": {
        "capturedAmount":"0",
        "remainingAmountTocapture":"0",
        "refundedAmount":"1200",
        "remainingAmountToRefund":"0"
    },
    "transactionLogHistory": [{
        "timeStamp": "",
        "operation": "",
        "amount": "",
        "transactionId":"",
        "transactionText"
        "requestId": ""
    }]
}
```

7.7.7 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
remainingAmoutToCapture	Integer	No	Total remaining amount to capture
refundedAmount	Integer	No	Total refunded amount of the order
remainingAmountToRefund	Integer	No	Total remaining amount to refund

7.7.8 Description

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. Potential operations are described in section 4.8
amount	Integer	No	Amount performed on the given operation
transactionId	String	No	Vipps Trajnsaction 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

7.7.9 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

7.8 Get Order Status

7.8.1 Overview

The API 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.

7.8.2 URL

```
v1/payments/{orderId}/serialNumber/{merchantSerialNumber}/status
```

7.8.3 Method

```
GET
```

7.8.4 URLParams

```
orderId =[Integer | String]    REQUIRED
merchantSerialNumber =[Integer | String]   REQUIRED
```

7.8.5 Success Response

Http Status Code	Content
200	ok

7.8.6 Response Body

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

}
```

7.8.7 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 4.7