Vipps – Invoice API

Version: **2.4**
Date produced: **11-Apr-2018**

Content

Terminology

Term	Description
Consumer	The end user of the Vipps mobile application
Invoice Merchant (Issuer)	The organization issuing the invoices
Merchant	The organization issuing the invoices
Invoice Hotel	An entity (e.g. ERP) that is integrated towards Vipps and has an HotelID
Customer	The end user of the Vipps mobile application
Direct capture	The amount is directly withdrawn from Customers account
KID	A KID number (kundeidentifikasjonsnummer) used by the payment of a bill to identify the customer and the invoice

Overview

Vipps Invoice APIs enables you to distribute invoices via Vipps enabling customers to view and pay invoices in Vipps app. Vipps also reminds the consumers through interactive notification on Invoices that are nearing due-date.

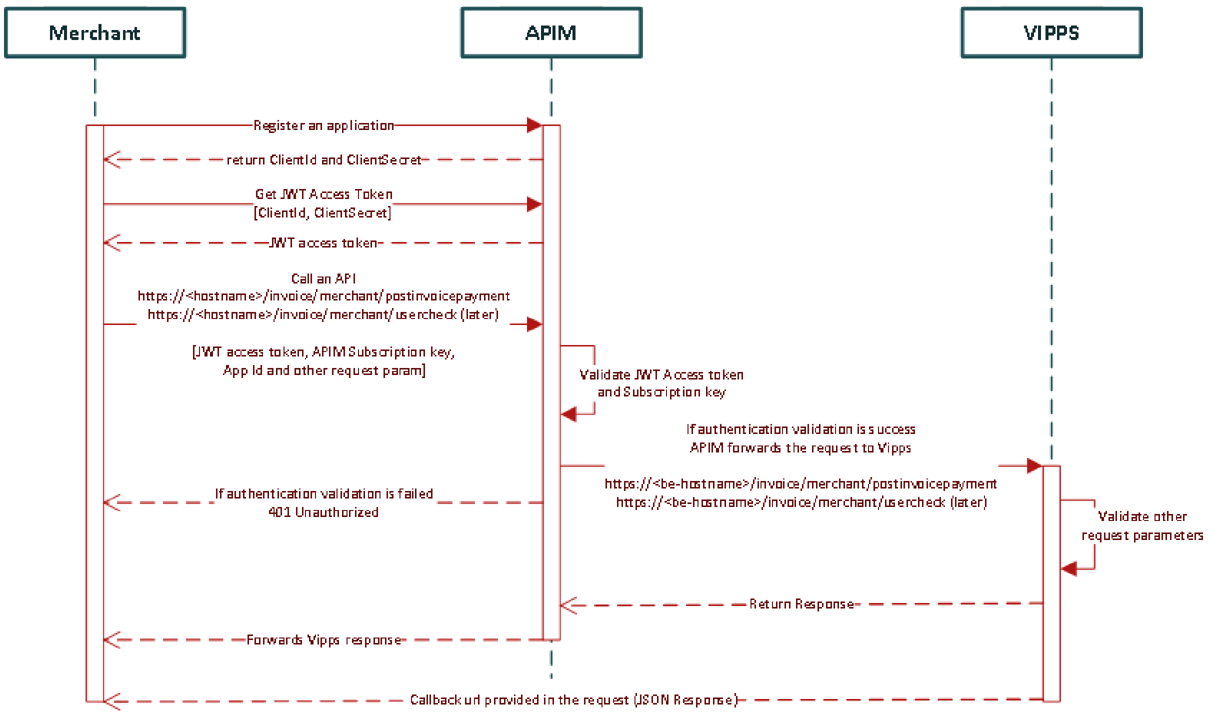
Target Audience

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

1. API Calls Flow

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 Invoice merchant (Issuer/Hotel) 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 https://<hostname>/invoice/merchant/postinvoicepayment https://<hostname>/invoice/merchant/usercheck (later)
[JWT access token, APIM Subscription key, App Id and other request param]
Validate JWT Access token and Subscription key
If authentication validation is success APIM forwards the request to Vipps
https://<be-hostname>/invoice/merchant/postinvoicepayment https://<be-hostname>/invoice/merchant/usercheck (later)
If authentication validation is failed 401 Unauthorized
Validate other request parameters
Return Response
Forwards Vipps response
Callback url provided in the request (JSON Response)

All communication with the Vipps Invoice 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 to authenticate all subsequent calls to Vipps API. See table 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

2. Response codes

Vipps Invoicing 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), 5xx are the Vipps internal errors.

Once a request is sent a receipt is returned containing a response code and a receipt ID.

2.1 Success Codes

```
200 - OK (HTTP)
9000 - Success
```

```
{
  "invoiceHotelInfo": {
    "invoiceHotelId": xxxx
  },
  "merchantInfo": {
    "merchantSerialNumber": "xxxx"
  },
  "receipt": 10103,
  "response": {
    "responseCode": "9000",
    "responseMessage": "SUCCESS"
  }
}
```

2.2 Error Codes

```
400 - Bad request (Missing a required parameter or Bad request formats)
401 - Unauthorized
403 - Request Forbidden
404 - Resource Not Found
405 - Request method not supported
415 - Unsupported 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:

```
{
 "response": {
    "responseCode": "1002",
    "responseMessage": "Request already received."
  }
}
```

field	example	description
responseCode	1001	error code which uniquely identifies an error scenario
responseMessage	"Authentication Failed"	Error message to display

2.3 Error Representation

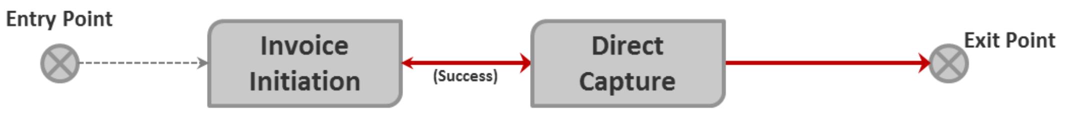
Error Groups	Range	Description
Authentication	1001	Authentication Failure because of wrong credentials provided
Idempotency	1002	Request already received
InvalidRequest	N/A	Request contains invalid parameters; invalid field name will be error code
VippsError	5001	Internal Vipps application error
Customer	1003-1009	Error raised because of Vipps user (Example: User not registered with Vipps..)
Merchant	2001-2999	Errors regarding the merchant (Invoice issuer)

2.4 Error codes

Error Group	Error Code	Error Message
Authentication	1001	Provided credentials doesn't match
Idempotency	1002	Request already received
InvalidRequest	{field_name will be the error code}	Description about what exactly the field error is
VippsError	5001	Description about the internal error
Customer	1003	User not registered or not active in Vipps
Customer	1004	Mobile number is not valid
Merchant	2001	Number of invoice requests has been exceeded
Merchant	2002	Number of customers has been exceeded
Merchant	2003	Merchant not available or deactivated or blocked
Merchant	2004	Invoice already exist

3. Payment flow

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



3.1 Invoice Initiation

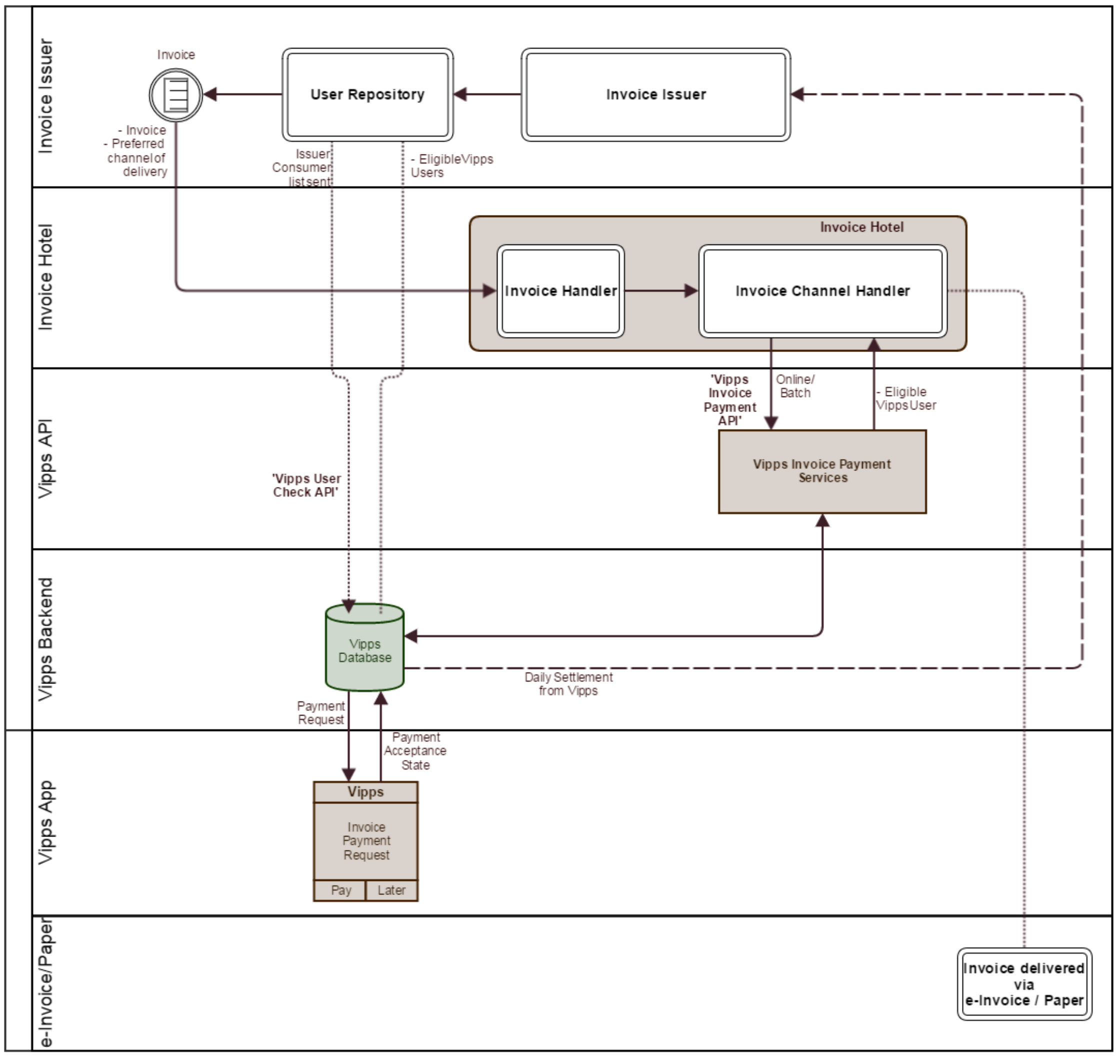
First call in payment flow initiates Invoice payment request that is subject of customer (end user) confirmation. Once Invoice is posted through the API, a callback will be made against the URL for posting the status of the delivered Invoice request back to intended Users. Payment has status *Initiated* and customer is notified about Invoice request in mobile app. If customer doesn't confirm, the Invoice request will expire after the Due Date (+ Grace Period).

4. Exception handling

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 measures are described briefly in its own section for example payment request. In this chapter we are describe how the merchant should handle when exception occurs.

4.1 Exception scenarios

The most critical action in Invoice flow is when the Invoice Payment service call is evoked. The flow diagram below shows that to successfully fulfill the 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.

4.1.1 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.

4.1.2 Callback aborted/interrupted

If the communication is broken after payment request is processed and Vipps is unable to execute callback, the callback transaction will be retired after a certain time. Please note that callback can be executed at any time within timeframe of 24 hours after Invoice payment request is sent, although typically the callback is handled within 4 hours. In other words, if the merchant doesn't receive any confirmation on payment request call within callback timeframe, the request should be treated as aborted. In case if callback failed at first attempt, we try 3 callback attempts (CallBackRetry Job timings - 4,6,8,15,18). If still the callback is not success, we abort that request. So you should wait for 24 hours and then try the request again.

5. Access Token

5.1 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.

5.2 URL

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

5.3 Method

POST

5.4 Request Headers

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

5.5 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

5.6 Success Response

Http Status Code	Content
200	OK

5.7 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_rCnW1cr3m7RfEl6z8e5VQva9A
Ot4OG5NuIrLJqmhb3KHBa3GusK6KLf-pVjF6fnS5r0ZQ5foP-VqOCiK9CUUATjHEOflgy1ubvA"
}
```

5.8 Response 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

5.9 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

5.10 Error Response Body

5.10.1 400 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"
}
```

5.10.2 401 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"
}
```

6. Invoice API definition

The Invoice API request allows invoice integration partner to initiate Vipps Invoice payment flow. Once a request is sent a receipt is returned containing a response code and a receipt ID.

Vipps process the API calls through batch jobs which will execute callback to the registered URL, sent as part of the request, with the status of the request. The callback call will be made via HTTPS, without any credentials. The batch jobs also distribute the invoice if the request is correct.

Each invoice payment request is uniquely identified by composite of *HotelId*, *merchantSerialNumber* and *InvoiceId*. In order to identify sales channel payments are coming from, a *merchantSerialNumber (To identify issuer) and HotelID (To identify Invoice Hotel)* are used to distinguish between them.

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

After the consumer has confirmed payment in the application, Vipps will execute direct capture of funds on the customer card used for the transaction.

6.1 Request Headers

Header Name	Header Value	Optional	Description
Authorization	“Bearer <jwt access token>”	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
Accept-Language	no	Yes	Allowed languages at present is Norwegian
X-TimeStamp	Time stamp when the request called	Yes	Time to call
X-Source-Address	Either source ip address or device id and terminal id for mobile application	Yes	Identifying the request source
X-Request-Id	To identify the idempotent request	No	For making request to be idempotent this ID is mandatory. Needs to be numeric and max 30 characters.

Ocp-Apim-Subscription-Key	Base 64 encoded string	No	Subscription key for Invoice product. This can be found in User Profile page on Vipps developer portal

```
Content-Type:application/json
Accept-Language:en
Authorization:Bearer xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
X-TimeStamp:01.08.2016 12:12
X-Request-id:1234567890987654321
Ocp-Apim-Subscription-Key:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

6.2 URL

`https://<hostname>/Invoice/invoicing/requestPayment`

6.3 Method

`POST`

6.4 Request Body

```
{
    "invoiceHotelInfo": {
        "invoiceHotelId": XXXX
    },
    "merchantInfo": {
        "merchantSerialNumber": "123456"
    },
    "numberOfInvoice": 2,
    "invoices": [{
        "mobileNumber": "90090900",
        "invoiceId": "123456789-123456789-123456789",
        "reference": "12345678909876543 21",
        "amount": 1200,
        "dueDate": "01.12.2016 17:38:17",
        "invoiceURL": "https://documentURL.com",
        "Text": "Sample text",
        "timeStamp":"2016-12-01T07:07:07-02:00"
    },
     {
        "mobileNumber": "90090901",
        "invoiceId": "123456789-123456789-123456790",
        "reference": "2345678909876543 23",
        "amount": 1200,
        "dueDate": "01.12.2016 17:38:17",
        "invoiceURL": "https://documentURL.com",
        "Text": "Sample text",
        "timeStamp":"2016-12-01T07:07:07-02:00"
    }],
    "callback" : "https://callback.com"
}
```

Id	Type	Size	Optional	Description
invoiceHotelId	Integer	-	No	Unique identifier of Invoice Hotel, generated by DB sequence
merchantSerialNumber	String	6	No	Identifies a merchant sales channel (Invoice Issuer) i.e. invoicing etc.
callback	String	255	No	Vipps will use this callback URL to POST the response describing how the request was processed. The URL needs to respond 200 OK to the POST or callback will fail and aborting the invoice processing. Length of the string is 255 characters.
numberOfInvoice	Integer	-	No	Number of invoices in the request, there will be a max check on number of invoice sent in a request.
mobileNumber	String	8-12	No	Mobile number of the user who has to pay for the transaction from Vipps. Allowed format: +47-xxxxxxxx +47xxxxxxxx xxxxxxxx
invoiceId	String	50	No	Id which uniquely identifies an Invoice. Maximum length is 50 alphanumeric characters
referance	String	30	No	KID Id, this will be available for some specific invoice
amount	Integer	-	No	Amount in øre. 32 Bit Integer (2147483647)
Text	String	100	No	Transaction text that can be displayed to end user
dueDate	String	-	No	last date to pay a particular invoice, in dd.MM.yyyy HH.mm.ss
invoiceURL	String	255	No	Link to access invoice from Vipps app See creating InvoiceURL section for details.
timeStamp	String	-	No	Timestamp in ISO-8601 representing when the order has been made by merchant

6.5 Success Response

Http Status Code	Content
200	OK

6.6 Response Body

```
{
  "invoiceHotelInfo": {
       "invoiceHotelId": 100001
   },
   "merchantInfo": {
           "merchantSerialNumber": "123456"
       },
  "receipt": 1234353,
  "response": {
       "responseCode": 9000 | errorCode,
       "responseMessage":"SUCCESS" | "errorMessage"
   },


}
```

6.7 Description

Id	Type	Size	Optional	Description
invoiceHotelId	Integer	5	No	Unique identifier of Invoice Hotel, generated by DB sequence
merchantSerialNumber	String	6	No	Identifies a merchant sales channel (Invoice Issuer) i.e. invoicing etc.
receipt	Integer	15	No	Receipt Id for a request, this is a unique identifier for a request
responseCode	Integer	4	No	Response code of acknowledgement, i.e 9000 or error code
responseMessage	String	255	No	Response message of acknowledgement, i.e Success or error message

6.8 Error Response

Error Code	Content	Description
401 (HTTP)	Unauthorized	When Access token is invalid or expired
1001	Authentication failure	Authentication failed because of invalid credentials
1002	Idempotency Failure	Request already received
Invalid request field	Invalid Request	Invalid request field will be error code
2001	Merchant Error code	Number of invoice requests has been exceeded
5001	Internal Server Error	Some internal error occurred in Vipps
2003	Merchant	Merchant not available or deactivated or blocked
404(HTTP)	Resource not found	Resource not found
403(HTTP)	Request forbidden	Request forbidden due to lack of permission

6.9 Callback URL

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


Example:
POST https://callback.com/invoice/receipt/status
Content-Type: application/json
```

6.10 Callback Request Object

Response for valid Invoice Issuer:

```
 {
    "merchantInfo": {
        "merchantSerialNumber": "123456"
    },
    "receipt": 123453,
    "numberOfInvoicesReceived": 2,
    "numberOfInvoicesReceivedWithError": 1,
    "requestId": 1234566789,
    "errors": [
        {
          "invoiceid": "121212",
           "errorCode": "1003",
           "errorMessage": " User not registered or not active in Vipps",
        }
     ], "globalErrors" : {}
}
```

Response for invalid Invoice Issuer:

```
{
    "merchantInfo": {
        "merchantSerialNumber": "NSBWSHP12"
    },
    "receipt": 123453,
    "numberOfInvoicesReceived": 10,
    "numberOfInvoicesReceivedWithError": 10,
    "requestId": 1234567890987654321,
    "errors": [
        ],
     "globalErrors": {
              "globalErrorCode": "2003",
              "globalErrorMessage": "Merchant not available or deactivated
or blocked"
      }
}
```

6.11 Callback Request Object Description

Id	Type	Size	Optional	Description
merchantSerialNumber	String	6	No	Identifies a merchant sales channel (Invoice Issuer) i.e. invoicing etc.
receipt	Integer	15	No	Receipt Id for a request, this is a unique identifier for a request, sequence generated by Vipps system
numberOfInvoicesReceived	Integer	-	No	Number of invoices received in this request from Invoice Hotel. Note that this is the number given as numberofInvoices in the request, it is not validated against the actual of invoices in the request.
numberOfInvoicesReceivedWithError	Integer	-	No	Number of invoices are failed in validation
invoiceId	String	50	No	Id which uniquely identifies an Invoice. Maximum length is 50 alphanumeric characters
errorMessage	String	255	No	Part of Errors section which contains errors related to each invoice. Error message describing the reason for failure
errorCode	String	255	No	Part of Errors section which contains errors related to each invoice. In case error is type of Invalid Request filed, ex: MERCHANT_INFO_MISSING
globalErrorMessage	String	255	No	Global error message
globalErrorCode	String	255	No	Global error code
Request ID	Integer	255	No	Same as X-Request ID in the request header

6.12 Callback Response

Http Status Code	Content
200	OK
404	Resource not found
403	Request forbidden
400	Bad request
415	Unsupported media type
405	Unsupported request type

6.13 Invoice URL Creation Logic from Vipps

There is a secure URL generation technique implemented in Vipps, where the original invoice URL is appended with signature (hash value generated by pre-shared secret),ttl, InvoiceId, invoiceTimeStamp. When user taps to see original invoice, original invoice URL gets changed to secure invoice URL and send to URL end point.

Invoice Hotel validates the URL by creating signature with preshared secret on their end and returns the original invoice in case of successful validation. The shared secret is provided by Vipps together with production credentials.

To get the invoice details "Post" the invoice URL appended with "identifier" together with the request parameters and the "signature". Supported parameters are timestamp, ttl and invoiceId
We use the following to create the signature append the values for timestamp, ttl, InvoiceId and secret. Then create a hexadecimal message digest (hash value) of the string using the SHA-1 function

Keywords	Description	Value
client		**secretkey**
secretkey	constant value	vinvoice
timestamp		System.currentTimeMillis() / 1000
ttl	constant value	600
invoiceId	Invoice Id from request	
signature		DigestUtil.getShaHex(**timestamp** + **ttl** + **invoiceId** + **secretVal**)
secretVal	Constant value	Shared Secret

Example for creating URL:

```
/*
https://documentBaseURL.com/vipps/view_invoice/vinvoice/?timestamp=1480576027&ttl=600&invoiceId=abcd-1234-abcd-1234-abcd-1234&signature=987654321x987654321x987654321
*/

// Shared secret between Integration Partners and Vipps
HashMap<String, String> userSecrets = new HashMap<String, String>()
{{
        put("vinvoice", "Shared secret value");
}};
String identifier = "vinvoice";
String secret = userSecrets.get(identifier);
String invoiceId = "abcd-1234-abcd-1234-abcd-1234"; // invoiceId received from API
String invoiceURL= "https://documentBaseURL.com/vipps/view_invoice/vinvoice/" ; // invoiceURL received from API
long timestamp = (System.currentTimeMillis() / 1000);
long ttl=600; // URI valid for 10 minutes, hardcoded in the system

// Generate sha1 hash of timestamp, invoiceId and secret.
String hashResult = org.apache.commons.codec.digest.DigestUtils.sha1Hex(timestamp + ttl+ invoiceId + secret);
StringBuilder testUrl = new StringBuilder();
testUrl.append(invoiceURL);
testUrl.append(identifier);
testUrl.append("/");
testUrl.append("?timestamp=" + timestamp);
testUrl.append("?ttl=" + ttl);
testUrl.append("&invoiceId=" + invoiceId);
testUrl.append("&signature=" + hashResult);
System.out.println(testUrl.toString());// Print working link
```

6.14 Invoice URL validation Logic from Vipps

```
/*
URI to validate
https:// documentBaseURL.com/vipps/view_invoice/?invoiceId=abcd-1234-abcd-1234-abcd-
1234&timestamp=1480576027&ttl=600&signature=987654321x987654321x987654321
*/

// Shared secret between Invoice hotel and Vipps
String secret = "Shared secret value";
String invoiceId = request.getParameter("invoiceId");
String timestamp= request.getParameter("timestamp");
String ttl= request.getParameter("ttl");
String signature= request.getParameter("signature");

// Generate sha1 hash of timestamp, time to live, invoiceId and secret.
String hashResult = org.apache.commons.codec.digest.DigestUtils.sha1Hex(timestamp + ttl+
invoiceId + secret);
return (  Long.parseLong(timestamp) + ttl  >= (System.currentTimeMillis() / 1000)  &&
hashResult.equals(signature) );
```

6.15 Service Capacity

Service	Maximum Load	Exceeding Maximum Capacity
Sending Invoices through the API in a single call	500	Error will be returned with ‘Limited exceeded’

7. OCR settlement Files through MFT

The MFT (Managed file transfer) solution is a DNB System used for receiving and distributing files in a controlled and secure matter. The MFT solution provides the OCR settlement file which contains information regarding the invoices that has been paid by the consumer.

7.1 DNB sFTP server supports

- Version 5 sFTP protocol, as supported by OpenSSH
- Inbound scp commands using SSH / SCP protocol, as supported by OpenSSH
- The sFTP server will not allow you to set permissions or change attributes on the file on our side.

7.2 How to retrieve files

Files from DNB will be found in the folder called "inbox".

The automation of file transfer depends on which file transfer client that is used. Please use file transfer client's help function.

7.3 Naming Convention

The Settlement (OCR) file will have the following naming convention:

OCRV<date of settlement in yyyyMMdd format>.<MerchantSerialNumber>.txt

For example: OCRV20170114.110023.txt

7.4 Example file

OCRV20170114.Sample.txt

7.5 OCR specification

VippsInvoice_OCR_SPEC_April2017.xlsx

7.6 How to get a file transfer client

Various SFTP clients can be acquired either as freeware from the net, or as licensed software. There are SFTP clients for all common operating systems like Linux, Unix, Windows and macOS.

The rest of the documentation is intended as an initial help to get started with SFTP from a Windows platform.

Link to freeware for Windows that supports SFTP (There are others that can be used):

WinSCP http://winscp.net/eng/index.php

7.7 Access to DNB MFT Infrastructure

To access MFT a new or existing MFT account is needed. Once an account is created you will receive a MFT partner ID which is your user name.

After SFTP client is acquired and installed, a User Identity key pair (SSH-2 RSA type with 2048 key length, public and private key) for authentication of the user must be generated.

On Windows you can use the program PuTTYgen (provided with the installation of WinSCP) to generate the key pair. Start PuTTYgen and generate the SSH2 RSA key pair, length 2048

Save your private key and public key, the private key is used in the SFTP Client's Login configuration.

E-mail the Public key to: integration@vipps.no
Both test and production keys should be sent at the same time)

For operating systems other than Windows, consult the documentation for generation of key pairs. On Unix/Linux, the command is ssh-keygen.
Please secure the private key by using Passphrase or storing the private key on a location with limited access. The use of passphrase may cause automation problems.

8. Vipps Info Torg Data Query Service

All invoice issuers that would like to distribute their invoices through Vipps will need to know whom of their customers have agreed to receive invoices through Vipps. Therefore DNB/Vipps have established a Vipps Customer Data Query Service at infotorg for their Integrator Partners.

For the Data Query Online web service there are some restrictions. Each request can contain from 1 to 1000 customers. This means that larger customer loads (more than 1000) must be treated using a loop. For å larger amount of request at the same time, you will have til look in to the Batch service.

Please contact integration@vipps.no for the specifications of the different services.

8.1 Online web services

The online interface will be provided as a SOAP based web service. UTF-8 encoding is used for both SOAP and WSDL.

The integration partners must establish the web service integration in their systems themselves.

8.2 Batch services

The batch interface will be provided through ERVYs SFTP-Server "sftp.infotorg.no". Each Integration partner will be offered an account on this SFTP-server, with their own identification credentials.

8.3 Service Capacity

Service	Maximum Load	Exceeding Maximum Capacity
Number of customers to the Data Query Service	Online API – 1000 Batch - 2900000	The Customer Query will not be executed