CanPay RemotePay Integration Developer Guide

Version 1.4.0-dev

Table of Contents

CHANGE LOG

Version #	Change Details	Release Date
1.2.7	Introduction of Auth call before the void call. This impacts REST API to reverse/void a payment/transaction	05/04/2023
1.2.8	End Points updated for REST API to reverse/void a payment/transaction. The previous endpoint "/paymentreversal" is still active but will be deprecated soon. New endpoint is "/transactionreversal"	05/10/2023
1.2.9	In CanPay's Javascript file add new optional parameter named **merchant_order_id**	05/17/2023
1.3.0	In CanPay's Javascript file add new optional parameter named **return_consumer_given_tip_amount**	08/29/2023
1.4.0	In CanPay's Javascript file add new optional parameters named **delivery_fee** and **split_funding_merchant_id.** Endpoint updated for Call REST API to get intent_id. New endpoint added Call REST API to get a split funding merchant list Response updated for Call REST API to get details of a payment/transaction, Call REST API to eCommerce Dashboard Data, Call REST API to eCommerce Dashboard load more data and Call REST API for Dashboard Advanced search	01/28/2025

JS Web Checkout

Introduction

Start accepting payments from customers on your Merchant/Integrator website or mobile app using CanPay JS Web Checkout. This checkout method requires minimum development effort.

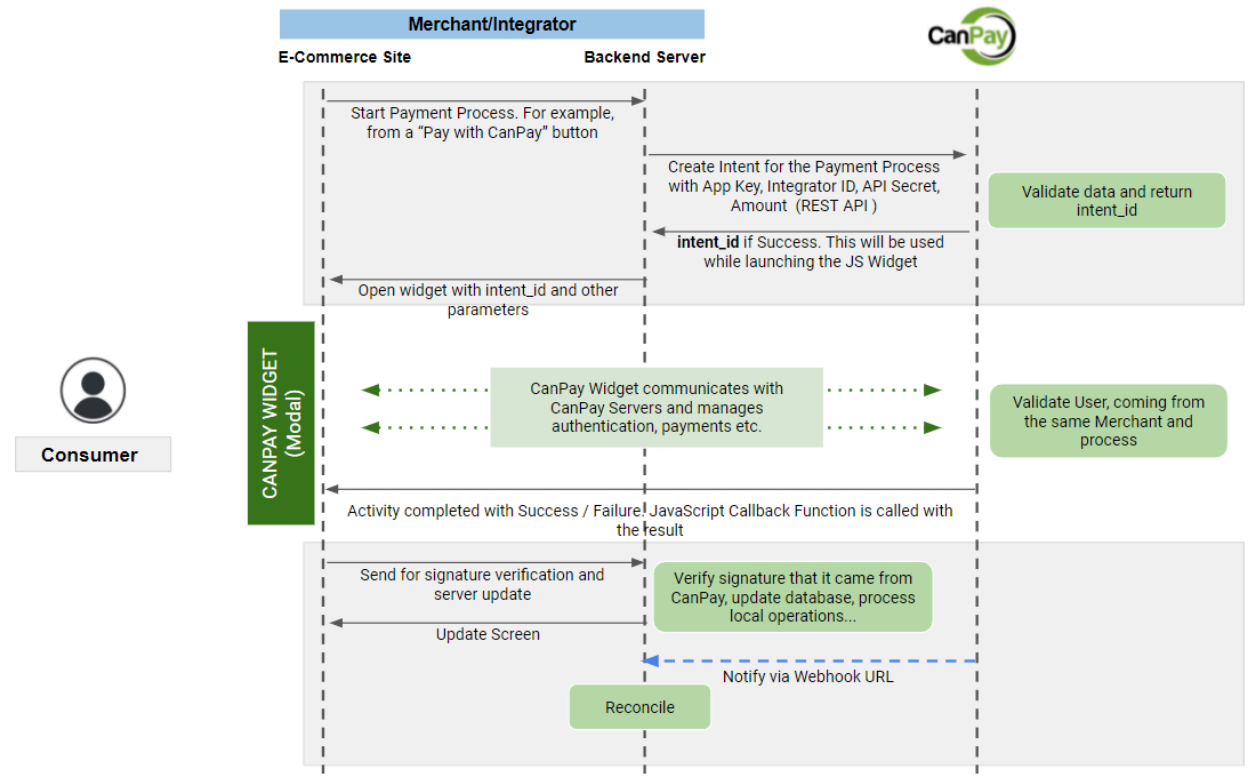
CanPay has developed the JS Checkout method and manages it. As a CanPay Merchant/Integrator you can easily integrate with CanPay via a JavaScript widget and a few API calls.

This document explains how Merchants/Integrators can initiate a CanPay payment from your website, marketplace, or mobile application and allow your customers to link their CanPay account for easy future passwordless payment.

Prerequisites

- Get **App Key**, **Integrator ID**, **API Secret, canpay_internal_version** and **Sandbox URL** for development and testing
- Provide a **Webhook URL** to receive important events like payment notifications
- Get code samples for front-end (JavaScript) and future server-side libraries from CanPay.

Overview of the Integration


Merchant/Integrator
E-Commerce Site
Backend Server
CanPay
Start Payment Process. For example, from a "Pay with CanPay" button
Create Intent for the Payment Process with App Key, Integrator ID, API Secret, Amount (REST API)
Validate data and return intent_id
intent_id if Success. This will be used while launching the JS Widget
Open widget with intent_id and other parameters
CANPAY WIDGET (Modal)
Consumer
CanPay Widget communicates with CanPay Servers and manages authentication, payments etc.
Validate User, coming from the same Merchant and process
Activity completed with Success / Failure. JavaScript Callback Function is called with the result
Send for signature verification and server update
Verify signature that it came from CanPay, update database, process local operations...
Update Screen
Notify via Webhook URL
Reconcile

Integration Steps

These are the high-level integration steps that Merchant/Integrator needs to follow:

- Add App Key, Integrator ID, API Secret, canpay_internal_version and Sandbox URL to system configuration. These parameters are required for processing payments through CanPay.
- Include necessary JavaScript in the application. This Javascript file will be hosted by CanPay.
- Prepare code to call the CanPay REST APIs from Integrator's backend server to get the one time use **intent_id.** This **intent_id** will expire once the payment is complete or the payment amount changes. For each payment, a new **intent_id** needs to be created with the transaction amount. This API is called from the server so that the API Secret is never passed to the frontend application.
- There are a total of Ten (3 optional for Subscription based payment) REST APIs to call for simple integration:

  - https://sandbox-api.canpaydebit.com/integrator/authorize
  - https://sandbox-api.canpaydebit.com/integrator/transactiondetails
  - https://sandbox-api.canpaydebit.com/integrator/transactionreversal
  - https://sandbox-api.canpaydebit.com/integrator/getmodificationreason
  - https://sandbox-api.canpaydebit.com/integrator/transactionupdate
  - https://sandbox-api.canpaydebit.com/integrator/modificationhistory
  - https://sandbox-api.canpaydebit.com/integrator/dashboard
  - https://sandbox-api.canpaydebit.com/integrator/loadmoretransactions
  - https://sandbox-api.canpaydebit.com/integrator/advancesearchtransactions
  - https://sandbox-api.canpaydebit.com/integrator/acceptpayment
  - https://sandbox-api.canpaydebit.com/integrator/add-edit-plan
  - https://sandbox-api.canpaydebit.com/integrator/subscriptionupdate
  - https://sandbox-api.canpaydebit.com/integrator/shipping-status-update
- Add the necessary Javascript code to the application to launch the CanPay Payment widget
- Implement the Javascript callback function. This callback function will be called by the JS widget to pass results back to the parent site once the payment process finishes.
- Implement the Webhook (optional). This is to get information from the CanPay server on the completion of the payment activity.
- Test against the Sandbox URLs

Building an Integration

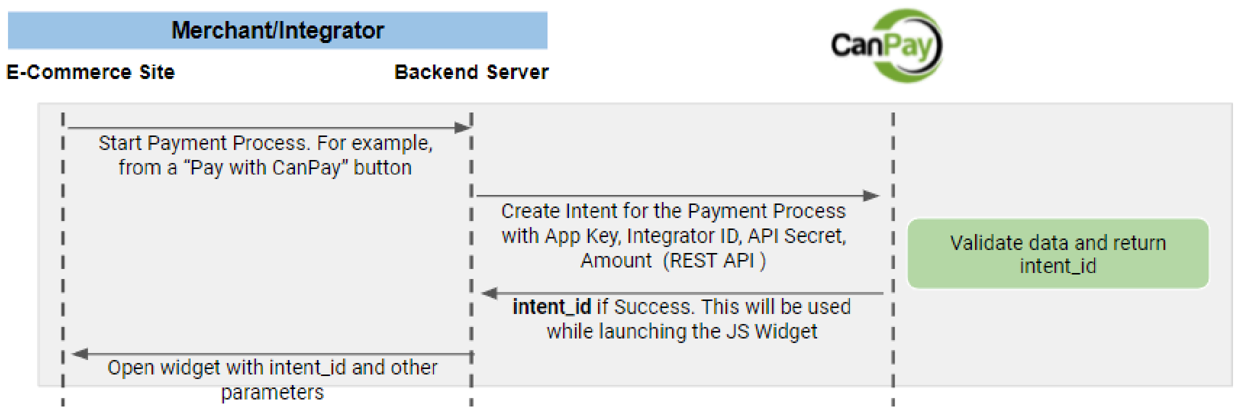
Add Credentials

Add App Key, Integrator ID, API Secret and Sandbox URL to system configuration. These parameters are required for processing payments through CanPay.

WARNING: Never send API Secret to the frontend JavaScript code or other parties except for CanPay REST APIs.

Start processing a Payment

Launching the CanPay payment widget is a two-step activity as shown below.


Merchant/Integrator
E-Commerce Site
Backend Server
CanPay
Start Payment Process. For example, from a "Pay with CanPay" button
Create Intent for the Payment Process with App Key, Integrator ID, API Secret, Amount (REST API)
Validate data and return intent_id
intent_id if Success. This will be used while launching the JS Widget
Open widget with intent_id and other parameters

Step 1: Call REST API to get intent_id

Once a consumer clicks on the CanPay payment button, the Merchant/Integrator needs to make a call to the CanPay server from the site backend to get an **intent_id**. This is done to avoid sending API Secret to the frontend code.

API Sample

```
curl --location --request POST 'https://sandbox-api.canpaydebit.com/integrator/authorize' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--form 'app_key="sandbox_key_a1sdkds2"' \
--form 'api_secret="xxxxxxxx"' \
--form 'integrator_id="gxt40fob"' \
--form 'canpay_internal_version="bh6y098k"' \
--form auth_only="true"'\
--form 'amount="10"'\
--form 'delivery_fee="2"' \
--form 'split_funding_merchant_id="24272d7136c2ca522a14ca8512a32198"' \
```

Parameters

- **app_key (Required):** Will be provided after application approval and importing vendor details.
- **api_secret (Required):** Will be provided after final testing and importing vendor details.
- **integrator_id (Required):** Will be provided after final testing and importing vendor details.
- **canpay_internal_version(Required):** Will be provided by CanPay Admin. This will be used to maintain our internal version.
- **auth_only (Optional):** If you send it with value true then the '**amount**' will be optional. Else amount will be required.
- **amount (Required if auth_only if not set or false)**: Need to send the transaction amount as we will check this amount later to prevent fraudulent activities.
- **delivery_fee (Optional):** Provide this if a delivery fee is applicable for this transaction.
- **split_funding_merchant_id(Optional):** Provide this if a delivery fee is applicable for this transaction. You will get this value from the API : Call REST API to get a split funding merchant list

Response

On Success:

```
{
    "code": 200,
    "message": "Authenticated successfully.",
    "data": {
        "intent_id":
```

```
"adb731e81e0a2e5c18c82c0e541cddfd52419b54da8d24df4549fe02b00ac911"
    }
}
```

On Error:

```
{
    "code": 599,
    "message": "Unauthorized Merchant.",
    "data": null
}
```

Note: This intent_id must be stored in the database against the checkout/payment on the Merchant/Integrator application. This intent_id will be used for future validation/reconciliation and voiding a transaction.
Also, add the 'User-Agent' in the header with the necessary value of the library you are using. E.g: If you are using CURL then use 'user-agent: curl/7.77.0' or if you are using Postman then use
'user-agent: PostmanRuntime/7.29.0'.

Step 2 - Launch the Widget with the intent_id

Add CanPay's Javascript file to the page in the head section.

```
<head>
    <!-- other html/js code -->
    <!-- Include CanPay JS in head -->
    <script type="text/javascript"
src="https://sandbox-remotepay.canpaydebit.com/cp-min.js"></script>
</head>

<body>
    <!-- other html/js code -->
</body>
<script>
    /* Define initialization parameters */
    var config = {
        "intent_id":
```

```
"adb731e81e0a2e5c18c82c0e541cddfd52419b54da8d24df4549fe02b00ac911",
        /* intent_id received from REST API call */
        "subscription_payment": 1,
        /* Integrator needs to send 1 when there is subscription */
        "shipping_address": "Haven Road Oliver Springs, TN 37840",
        /* Integrator needs to send product shipping address when there is
subscription*/
        "plan_identifier": "basic01",
        /* If the has_subscribed parameter is present and has value 1 then
the plan_identifier parameter will be required and we will proceed further
with a subscription. Else it will work as it is now working for remotepay */

        "amount": "10.00",
        /* Transaction amount */
        "tip_amount": "0.00",
        /* Tip amount (optional) */
        "deivery_fee": "0.00",
        /* Delivery Fee (optional) */
        "split_funding_merchant_id": "”,
        /* Split Funding Merchant ID (optional) */
        "original_merchant": "",
        /* Original merchant(optional) */
        "is_guest": "",
        /* true/false. true for guest checkout */
        "need_modification_url": "",
        /* true/false. true for transaction modification URL */
        "return_consumer_given_tip_amount": "",
        /* true/false. true for integrator need consumer given tip amount */
        "passthrough": {},
        /* Any other info that needs to be passed to CanPay */
        "merchant_order_id": "cp-557",
        /* An Order ID that will identify the transaction in future
(optional) */
```

```
        "auth_id": "",
        /* Stored auth_id for passwordless payment (optional) */
        "login_callback": function(response) {
            /* This function receives the result of the login. Only useful
when login fails. Not useful for successful logins */
            console.log(response.message);
        },
        "link_callback": function(response) {
            /* This function receives the auth_id for future passwordless
logins. Store in the database against the user */
            my_function_store_at_server(response.data.auth_id) // implement
this function;
        },
        "processed_callback": function(response) {
            /* This function receives the result of the transaction. Call
server API for validation and further processing */
            console.log(response.data);
            /* This should print something like this:
            {
"response":
"{\"intent_id\":\"79af968a1f01813959a47a681f8c69500fb519e1252851a3e58aacc1d0
bdb3af\",\"canpay_transaction_number\":\"IVUBUZAJIFU9\",\"transaction_time\"
:\"2022-02-18T09:51:22.839521Z\",\"amount\":10.00,\"tip_amount\":0,\"passthr
ough_param\":\"\",\"expiry_date_for_book_now\":\"2022-09-15\",\"transaction_
modification_url\":\"https:\\/\\/sandbox-merchant-admin.canpaydebit.com\\/lo
gin\\/NDA0ODE1NzBiZjZhOTJhNDAzNTc4NzU3YTBhYjQ0N2IwMmU2MGNjMDg1ODQwMzM5NGY2Mj
UyMzljZWI0MWZmZg==\",\"merchant_order_id\":\"cp-557\"}",
"signature":
"54e5b59e3d93321aa0dd8f6ea61371c96e6e2c38cb43c8b3efb4abd32f1d77a6"
}
*/
/* Validate the signature of the response content on the server and process
further. response contains payment details for further processing like
updating transaction status etc. */
        },
        "intentId_validation_callback": function(response) {
            /* This function receives a message when Intent ID validation
fails. */
            console.log(response);
        },
    };
    /* Parameters defined. Launch the widget now */

<!-- Initiate CanPay JS widget. This is a sample implementation and
Merchant/Integrator need to instantiate and launch the widget on events
(like a checkout button) -->

    canpay.init(config);

/* . . . Callback functions will be triggered when the widget it on or
finishes */
</script>
```

Initialization Parameters

- **intent_id (Required):** Merchant/Integrator can find this id in the response from the authenticate API.
- **has_subscribed(Optional):** Integrator needs to send 1 when there is subscription. Else it will work as a normal one time payment.
- **shipping_address(Required if the has_subscribed is 1):** Integrator needs to send product shipping address when there is subscription.
- **plan_identifier(Required if the has_subscribed is 1):** If the has_subscribed parameter is present and has value 1 then the plan_identifier parameter will be required and we will proceed further with a subscription..
- **is_guest (Required):** If **customers do not want to create an account** or log in to purchase a product in the Merchant's e-commerce store, then **isGuest is true.** Otherwise, **isGuest is false**

- **need_modification_url(Optional):** If **Merchant/Integrator need transaction modification URL**, then **need_modification_url is true.**
- **return_consumer_given_tip_amount(Optional):** If **Merchant/Integrator need consumer given tip amount**, then **return_consumer_given_tip_amount is true.**
- **amount (Required):** This should be the total transaction amount and should match the value with the amount sent in the authenticate API.
- **tip_amount (Optional):** This should be the tip amount. Leave it blank if there is no tip.
- **original_merchant (Optional):** This is an extra parameter. The name provided will be used in the transaction response.
- **passthrough (Optional):** Extra optional parameter for transaction identification purposes. This should be JSON. This value will be sent back as is.
- **merchant_order_id(Optional):** Merchant Order ID paramete isr for the transaction identification purpose. If the merchant provides a value then the value will be sent back as is.
- **auth_id (Optional):** When the consumer selects to pay with CanPay for the first time the payment the auth_id will be null. Once the consumer links his/her account with that merchant/integrator, an auth_id will be provided from the response from the **link_callback** function. Merchant/Integrator is required to store this customer auth_id to support future passwordless purchases. The next time onwards Merchant/Integrator should send the **auth_id** so that the consumer is not required to sign into CanPay again.
- **login_callback (optional):** This function receives the login result. This is only useful when the login fails. Not useful for successful logins.
- **link_callback (optional):** Once the consumer links his/her account with Merchant/Integrator, this function receives the auth_id for future passwordless logins. This needs to be securely stored in the database against the user. In future passwordless payments, this auth_id needs to be passed. Pass null for guest checkout or where auth_id is null.
- **processed_callback:** This function receives the result of the transaction. Call server API for validation and further processing.
- **intentId_validation_callback (optional):** This function receives a message when Intent ID validation fails.

Verify Payment Details and Process

The **processed_callback** is called with two parameters: **response** and **signature**.

response contains the JSON string with the payment details and it looks like this:

<u>**Remote Payment**</u>:

```
{
"response":
"{\"intent_id\":\"79af968a1f01813959a47a681f8c69500fb519e1252851a3e58aacc1d
0bdb3af\",\"canpay_transaction_number\":\"IVUBUZAJIFU9\",\"transaction_time
\":\"2022-02-18T09:51:22.839521Z\",\"amount\":10.00,\"tip_amount\":0,\"deli
very_fee\":2,\"split_funding_merchant_id
\":"24272d7136c2ca522a14ca8512a32198",\"passthrough_param\":\"\",\"expiry_d
ate_for_book_now\":\"2022-09-15\",\"transaction_modification_url\":\"https:
\\/\\/sandbox-merchant-admin.canpaydebit.com\\/login\\/NDA0ODE1NzBiZjZhOTJh
NDAzNTc4NzU3YTBhYjQ0N2IwMmU2MGNjMDg1ODQwMzM5NGY2MjUyMzljZWI0MWZmZg==\",\"me
rchant_order_id\":\"cp-557\"}",
"signature":
"54e5b59e3d93321aa0dd8f6ea61371c96e6e2c38cb43c8b3efb4abd32f1d77a6"
}
```

<u>**Subscription Payment**</u>:

```
{
"response":
"{\"intent_id\":\"79af968a1f01813959a47a681f8c69500fb519e1252851a3e58aacc1d
0bdb3af\",\"canpay_transaction_number\":\"IVUBUZAJIFU9\",\"transaction_time
\":\"2022-02-18T09:51:22.839521Z\",\"amount\":\"10.00\",\"tip_amount\":\"0\",\"
passthrough_param\":\"\",\"consumer_identifier\":\"70352e68e57843fbb3be74b7
5ff43521\"}",
"signature":
"45e5b59e3d93321aa0dd8f6ea61371c96e6e2c38cb43c8b3efb4abd32f1d78d3"
}
```

signature contains the hash of the JSON string.

The content of the **response** field must be first validated with the signature to make sure the response is coming from the CanPay server.

Keyed Hash Message Authentication Code (HMAC) is used for creating the **signature**. The signature parameter contains the content of the response field SHA-256 hashed, with the API secret as the key.

On the server, the Merchant/Integrator needs to compute the same hash with the content of the data field and compare the result with the value of the signature. They must be equal.

Note: This must be done on the server as the API secret must not be included in the front-end code.

Reference implementations of the Keyed Hash Message Authentication Code (HMAC) are available in various languages under section HMAC Sample Code.

Once the signature is successfully verified, the JSON string with the payment details can be converted to JSON for further processing like updating transaction status, etc.

Details of Response Data

- **intent_id**: the intent_id used to initialize the Payment process
- **canpay_transaction_number**: An unique 12 character ID for the payment transaction
- **transaction_time**: Timestamp of when payment was made in UTC
- **amount**: Amount processed
- **tip_amount**: Tip amount processed
- **delivery_fee**: Delivery fee processed
- **split_funding_merchant_id**: The merchant id for split funding
- **passthrough_param**: The passthrough params passed to the Payment Widget
- **transaction_modification_url**: Transaction modification url (If **Merchant/Integrator** send **need_modification_url** as **true**)
- **merchant_order_id**: Merchant Order ID (If **Merchant/Integrator** provides the **merchant_order_id** value). Otherwise it will not be returned.
- **consumer_identifier**: Consumer Identifier.

Implement the Webhook (Optional)

Merchant/Integrator needs to implement a webhook to get events directly from the CanPay Server. Please see the section **Webhook Reference**.

Other REST APIs

Call REST API to get details of a payment/transaction

To fetch payment/transaction details, the Merchant/Integrator needs to make a call to the CanPay server from the application backend.

API Sample

```
curl --location --request POST
'https://sandbox-api.canpaydebit.com/integrator/transactiondetails' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--form
'intent_id="1b1637cceb55f9fe83d1b469d27253301b1122a055cda71b8a5cb2a10f253b93
"'
```

Parameters

- **Intent_id (Required):** Merchant/Integrator can find this id in the response from the authenticate API.

Response

On Success:

```
{
    "code": 200,
    "message": "Transaction details fetched successfully.",
    "data": {
        "transaction_number": "ZURIJU2UBI7I",
        "transaction_time": "2021-08-23 21:16:48",
        "amount": "2.00",
        "tip_amount": "0.00",
        "delivery_fee": "2.00",
        “split_funding_merchant”: “City Style Boutique”
        "updated_amount": "3.00",
        "status": "Success",
        "passthrough_param": null,
        "original_merchant_name": null,
        "expiry_date_for_book_now":  "2021-08-31"
    }
}
```

On Error:

```
{
    "code": 599,
    "message": "No transaction found on your given input.",
    "data": null
}
```

Probable status:

Pending: Pending for settlement	
Awaiting Consumer Approval: Transaction Modification request sent to consumer	
Voided: Canceled	
Approved By Consumer: When consumer accept modification or when decrease amount	
Declined By Consumer: When consumer declined modification	
Request Timeout: Transaction Modification time expired	
Settled: Already post to bank	
Success: Transaction successfully settled	

Call REST API to get a split funding merchant list

To fetch a split funding merchant list, the Merchant/Integrator needs to make a call to the CanPay server from the application backend.

API Sample

```
$ curl --location --request POST

 'https://sandbox-api.canpaydebit.com/integrator/getdeliverypartnerlist \

--header 'Accept: application/json' \

--header 'Content-Type: application/json' \

--form
'intent_id="1b1637cceb55f9fe83d1b469d27253301b1122a055cda71b8a5cb2a10f253b93"'
```

Parameters

- **Intent_id (Required):** Merchant/Integrator can find this id in the response from the authenticate API.

Response

On Success:

```
{
    "code": 200,
    "message": "Split funding merchants fetched successfully.",
    "data": [
        {
            "split_funding_merchant_id": "24272d7136c2ca522a14ca8512a32198",
            "split_funding_merchant_name": "City Style Boutique"
        }
    ]
}
```

Call REST API to reverse/void a payment/transaction

To reverse/void a payment/transaction details, Merchant/Integrator needs to make a call to the CanPay server from the application backend.

API Sample

```
$ curl --location --request POST
'https://sandbox-api.canpaydebit.com/integrator/transactionreversal' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--form
'intent_id="21e1f05565dfcfb89275a2dfdbb2536cb27485c5d11094cc75d2c7376e854f62
"' \
--form
'transaction_intent_id="a958a30e080c54199c822d2bf77e748d251ef126756b7bb3326d
0dbcdb10e227"'
```

Parameters

- **Intent_id (Required):** Merchant/Integrator can find this id in the response from the authenticate API with the **auth_only** value set to **true**.
- **transaction_intent_id(Required):** Merchant/Integrator can find this id returned as a response after executing the Pay Via CanPay API from the widgetResponse.

Response

On Success:

```
{
  "code": 200,
  "message": "Transaction voided successfully",
  "data": null
}
```

On Error:

```
{

    "code": 599,
    "message": <Probable error reasons are listed below>
    "data": null

}
```

Probable error reasons:

Intent ID is required.
Invalid Intent ID.
The Transaction Intent Id Field Is Required.
No Transactions found.
Transaction is not available for void.
Transaction already voided.

Call REST API to get transaction modification reasons

To fetch transaction modification reason list, the Merchant/Integrator needs to make a call to the CanPay server from the application backend.

API Sample

```
$ curl --location --request POST

'https://sandbox-api.canpaydebit.com/integrator/getmodificationreason' \

--header 'Accept: application/json' \

--header 'Content-Type: application/json' \

--form

'reason_type="increase"'
```

Parameters

- **reason_type(Required):** value should be “**increase**” or “**decrease**”.

Response

On Success:

```
{
  "code": 200,
  "message": "All Transaction Modification Reason fetched successfully.",
  "data": [
    {
      "id": "97096a465150f0fbdb8b9a36aeab7ec0",
      "reason": "Tax Increase",
      "need_approval": 1,
      "reason_type": "increase",
      "created_at": "2022-07-22 07:15:38",
      "updated_at": "2022-07-22 07:15:38"
    }
  ]
}
```

On Error:

```
{

    "code": 599,
    "message": "No reason found on your given input.",
    "data": null

}
```

Call REST API to Modify transaction

To modify a transaction, the Merchant/Integrator needs to make a call to the CanPay server from the application backend. Based on the reason and type some of the requests will be sent to the consumer for confirmation. We are going to call the webhook once we receive confirmation or rejection from the consumer to update the status.

API Sample

```
$ curl --location --request POST

'https://sandbox-api.canpaydebit.com/integrator/transactionupdate \

--header 'Accept: application/json' \

--header 'Content-Type: application/json' \

--form

'intent_id="a427c48e00280648031efcd8defb53de81bbbd826955d2b8599c4aa3f9ee06f38"

--form 'updated_amount="0.6"' \

--form 'reason_id="97096a465150f0fbdb8b9a36aeab7ec0"' \

--form 'reason_comment=""' \
```

Parameters

- **Intent_id (Required):** Merchant/Integrator can find this id returned as a response after executing the Pay Via CanPay API from the widgetResponse.
- **updated_amount (Required):** Requested amount you want to update.
- **reason_id (Required) :** Merchant/Integrator can find this id returned as a response after executing the get transaction modification reasons API. If the reason doesn't exist in the response then provide **'custom'** in place of the id. If you are using custom the **reason_comment** will be required.
- **reason_comment(Required if reason_id is 'custom'):**
  If you send reason_id as **'custom'** then you need to add a custom reason.

Response

On Success:

```
{
  "code": 200,
  "message": "Transaction modified successfully.",
  "data": "null"
}
```

```
{
  "code": 202,
  "message": "Transaction modification request sent successfully.",
  "data": "null"
}
```

On Error:

```
 {

     "code": 599,
     "message": <Probable decline reasons are listed below>
     "data": null

 }
```

Probable decline reasons:

No transaction found on your given input.
Transaction not modificationable.
Transaction not eCommerce type.
Transaction already settled.
Failed to generate QR code.
Database transaction failed.
Unknown User. Contact Administrator.

`Store for this terminal has been deactivated.`
`POS system for this device has been deactivated.`
`Invalid Terminal Type.`
`No account selected. Please select one account.`
`There is a problem with your bank. Please change your bank and try again.`
`Transaction Declined due to exceeding available Purchase Power.`
`Transaction declined due to low purchase power than the transaction amount.`
`Transaction Declined due to exceeding available Spending Limit.`
`Transaction Declined due to exceeding available Weekly Spending limit.`
`Please refresh the page or relink your bank account to proceed.`
`Payment Code has Expired. Customers must generate a new one to make a purchase.`

Call REST API to get history and current status of modified transaction

To fetch the history and current status of the modified transaction, the Merchant/Integrator needs to make a call to the CanPay server from the application backend.

API Sample

```
$ curl --location --request POST

'https://sandbox-api.canpaydebit.com/integrator/modificationhistory' \

--header 'Accept: application/json' \

--header 'Content-Type: application/json' \
--form
intent_id:1b1637cceb55f9fe83d1b469d27253301b1122a055cda71b8a5cb2a10f253b93
```

Parameters

- **Intent_id (Required):** Merchant/Integrator can find this id returned as a response after executing the Pay Via CanPay API from the widgetResponse.

Response

On Success:

```
{
  "code": 200,
  "message": "Modified transaction fetched successfully.",
  "data": [
    {
      "status": "Awaiting Consumer Approval",
      "reason": "Price Increase",
      "amount": "3.30",
      "local_transaction_time": "07-06-2022 05:39:10"
    },
    {
      "status": "Declined",
      "reason": "Price Increase",
      "amount": "3.00",
      "local_transaction_time": "07-06-2022 05:38:32"
    },
    {
      "status": "Approved",
      "reason": "Price Increase",
      "amount": "2.50",
      "local_transaction_time": "07-06-2022 05:35:16"
    },
    {
      "status": "Auto Approved",
      "reason": "Price Decrease",
      "amount": "1.00",
      "local_transaction_time": "07-06-2022 02:43:59"
    },
    {
      "status": "Initial",
      "reason": null,
      "amount": "1.40",
      "local_transaction_time": "07-06-2022 01:59:18"
    }
  ]
}
```

On Error:

```
{

    "code": 599,
    "message": "No transaction found on your given input.",
    "data": null

}
```

Call REST API to eCommerce Dashboard Data

To get eCommerce Dashboard data, the Merchant/Integrator needs to make a call to the CanPay server from the application backend.

API Sample

```
curl --location --request POST
'https://sandbox-api.canpaydebit.com/integrator/dashboard' \
--form
'intent_id="57034f4e71cea399ee090efa65eb123ab58329b4822202c4683dfec8b2649d1200"' \
--form 'from_date="2022-08-01"' \
--form 'to_date="2022-08-08"'
```

Parameters

- **Intent_id (Required):** Merchant/Integrator can find this id in the response from the authenticate API with the **auth_only** value set to **true**.
- **from_date (Optional)** : This should be the start date of the search criteria.
- **to_date (Optional)**: This should be the end date of the search criteria.

Note: If you skip the from_date and to_date parameter the system will return the last 7 days data. Also, don’t send the same date in from_date and to_date. To_date must be greater than from_date.

Response

On Success:

```
{
    "code": 200,
    "message": "Ecommerce Dashboard Data fetched successfully.",
    "data": {
        "number_of_transaction_to_expire": {
            "today": 0,
            "tomorrow": 0,
            "dayaftertomorrow": 0
        },
        "pre_load_transactions": {
            "pending_and_unpaid_transactions": [
                {
                    "intent_id":
"62bf402b7385e892acb50ba162fb7ee0a08feb2d2c2d666e4bbbce04a967faca",
                    "transaction_number": "EHE7IMIVITEF",
                    "orginal_amount": 1.79,
                    "orginal_tip_amount": 0,
                    "last_updated_tip_amount": 0,
                    "updated_amount": 0,
                    "transaction_date": "08-08-2022",
                    "status": "Pending",
                    "delivery_fee": 0,
                    "time_left_for_approval": "",
                    "expiry_date_for_book_now": "08-15-2022"
                },
                {
                    "intent_id":
"8755c185542ad5f629cc334e153d861bd6b601a999efe8490bba95b8267647fb",
                    "transaction_number": "E2EMECUYU5IQ",
                    "orginal_amount": 6,
                    "orginal_tip_amount": 0,
                    "last_updated_tip_amount": 0,
                    "updated_amount": 0,
                    "transaction_date": "08-08-2022",
                    "status": "Pending",
                    "delivery_fee": 0,
                    "time_left_for_approval": "",
                    "expiry_date_for_book_now": "08-15-2022"
                }
            ],
            "awaiting_consumer_approval_transactions": [
                {
                    "intent_id":
"05d64305a4e2c5717c7b1e39231cfa8e559ad382b534ab7089a8b89eda604074",
                    "transaction_number": "2EHU5UBU8U8A",
                    "orginal_amount": 1.8,
                    "orginal_tip_amount": 0,
                    "last_updated_tip_amount": 0,
                    "updated_amount": 1.9,
                    "transaction_date": "08-08-2022",
                    "status": "Awaiting Consumer Approval",
                    "delivery_fee": 0,
                    "time_left_for_approval": "17:09:51",
                    "expiry_date_for_book_now": "",
                    "approval_expiration_datetime": 1660024799,
                    "current_store_datetime": 1659963007
                }
            ],
            "approved_transactions": [
                {
                    "intent_id":
"cd3d77f038df1f6af3a87c3aa526b2e945038bc295eaa6c04c5f15f6e358dcfa",
                    "transaction_number": "0IPUFUHAXETI",
                    "orginal_amount": 1.2,
                    "orginal_tip_amount": 0,
                    "last_updated_tip_amount": 0,
                    "updated_amount": 2.2,
                    "transaction_date": "08-08-2022",
                    "status": "Approved",
                    "delivery_fee": 0,
                    "time_left_for_approval": "",
                    "expiry_date_for_book_now": "08-15-2022"
                },
                {
                    "intent_id":
"71dab54b697a0fa1d251761cb2ee0a076d2d593ea4e64669de6be9c5ae88c927",
                    "transaction_number": "I4EJIHUCUHA4",
                    "orginal_amount": 6,
                    "orginal_tip_amount": 0,
                    "last_updated_tip_amount": 0,
                    "updated_amount": 6.1,
                    "transaction_date": "08-04-2022",
                    "status": "Approved",
                    "delivery_fee": 0,
                    "time_left_for_approval": "",
                    "expiry_date_for_book_now": "08-11-2022"
                },
                {
                    "intent_id":
"f6de57e3e397551846ca9ab85ea4b5c8a435d32c270ceda1039d56e103bbcefa",
                    "transaction_number": "RUBU5ACA0AZA",
                    "orginal_amount": 2.5,
                    "orginal_tip_amount": 2,
                    "last_updated_tip_amount": 2,
                    "updated_amount": 1.5,
                    "transaction_date": "08-01-2022",
                    "status": "Approved",
                    "delivery_fee": 0,
                    "time_left_for_approval": "",
                    "expiry_date_for_book_now": ""
                },
                {
                    "intent_id":
"c923660086fbdb97ba29755817d5205a634b5b2bf9907169e2451a01eab4d1ab",
                    "transaction_number": "6E1UXUNEMUVE",
                    "orginal_amount": 2.3,
                    "orginal_tip_amount": 2,
                    "last_updated_tip_amount": 0.32,
                    "updated_amount": 4.3,
                    "transaction_date": "08-01-2022",
                    "status": "Approved",
                    "delivery_fee": 0,
                    "time_left_for_approval": "",
                    "expiry_date_for_book_now": ""
                }
            ],
            "declined_transactions": [
                {
                    "intent_id":
"13e81efd02a6b72e05ec8d712315b8fa3c6760f9e665cc420bdc44ed8e58fdbb",
                    "transaction_number": "LEPEDE4IRITE",
                    "orginal_amount": 1.1,
                    "orginal_tip_amount": 0,
                    "last_updated_tip_amount": 0,
                    "updated_amount": 2,
                    "transaction_date": "08-01-2022",
                    "status": "Consumer Declined",
                    "delivery_fee": 0,
                    "time_left_for_approval": "",
                    "expiry_date_for_book_now": ""
                }
            ]
        }
    }
}
```

Note: The Accept Payment button in your dashboard should appear only if the expiry_date_for_book_now is not null. If it is null that means that the transaction has already been marked as payment accepted. Also, if the store is not admin driven, the Accept Payment button will not appear in the dashboard. The time_left_for_approval is the time left for consumer approval.

On Error:

```
{
    "code": 599,
    "message": "Invalid Intent ID.",
    "data": null
}
```

Call REST API to eCommerce Dashboard load more data

To get eCommerce Dashboard all transactions data, the Merchant/Integrator needs to make a call to the CanPay server from the application backend.

API Sample

```
curl --location --request POST 'https://sandbox-api.canpaydebit.com/integrator/loadmoretransactions' \
--form 'intent_id="57034f4e71cea399ee090efa65eb123ab58329b4822202c4683dfec8b2649d12"' \
--form 'type="Approved"' \
--form 'from_date="2022-08-01"' \
--form 'to_date="2022-08-08"' \
--form 'limit="10"' \
--form 'page="1"'
```

Parameters

- **Intent_id (Required):** Merchant/Integrator can find this id in the response from the authenticate API with the **auth_only** value set to **true**.
- **type:(Required)** : Values should be => ‘**Pending And Unpaid**’ or ‘**Awaiting Consumer Approval**’ or **‘Approved’** or **‘Consumer Declined’**
- **from_date (Optional) :** This should be the start date of the search criteria.
- **to_date (Optional):** This should be the end date of the search criteria.
- **limit** : Number of items you wish to display per page
- **page**: Use to get next/previous page items. The page value Initial 1. To get next page items then increases by 1.To get previous page items then decreases by 1.

Example: If you have 30 records and you set the limit to 10. Now page 1 will return 10 records starting from 1. To get the next 10 records starting from 11 to 20 you should send the limit value as 10 and page value as 2. Don't change the limit value for next pages as it will lead to data mismatch.

Note: If you skip the from_date and to_date parameter the system will return the last 7 days data. Also, don’t send the same date in from_date and to_date. To_date must be greater than from_date.

Response

On Success:

```
{
  "code": 200,
  "message": "All transactions fetched successfully.",
  "data": {
    "current_page": 1,
    "data": [
      {
        "intent_id": "cd3d77f038df1f6af3a87c3aa526b2e945038bc295eaa6c04c5f15f6e358dcfa",
        "transaction_number": "0IPUFUHAXETI",
        "orginal_amount": 1.2,
        "orginal_tip_amount": 0,
        "last_updated_tip_amount": 0,
        "updated_amount": 2.2,
        "transaction_date": "08-08-2022",
        "status": "Approved",
        "delivery_fee" :  0,
        "time_left_for_approval": "",
        "expiry_date_for_book_now": "08-15-2022"
      },
      {
        "intent_id": "71dab54b697a0fa1d251761cb2ee0a076d2d593ea4e64669de6be9c5ae88c927",
        "transaction_number": "I4EJIHUCUHA4",
        "orginal_amount": 6,
        "orginal_tip_amount": 0,
        "last_updated_tip_amount": 0,
        "updated_amount": 6.1,
        "transaction_date": "08-04-2022",
        "status": "Approved",
        "delivery_fee" :  0,
        "time_left_for_approval": "",
        "expiry_date_for_book_now": "08-11-2022"
      },
      {
        "intent_id": "f6de57e3e397551846ca9ab85ea4b5c8a435d32c270ceda1039d56e103bbcefa",
        "transaction_number": "RUBU5ACA0AZA",
        "orginal_amount": 2.5,
        "orginal_tip_amount": 2,
        "last_updated_tip_amount": 2,
        "updated_amount": 1.5,
        "transaction_date": "08-01-2022",
        "status": "Approved",
        "delivery_fee": 0,
        "time_left_for_approval": "",
        "expiry_date_for_book_now": ""
    },
    {
        "intent_id": "c923660086fbdb97ba29755817d5205a634b5b2bf9907169e2451a01eab4d1ab",
        "transaction_number": "6E1UXUNEMUVE",
        "orginal_amount": 2.3,
        "orginal_tip_amount": 2,
        "last_updated_tip_amount": 0.32,
        "updated_amount": 4.3,
        "transaction_date": "08-01-2022",
        "status": "Approved",
        "delivery_fee": 0,
        "time_left_for_approval": "",
        "expiry_date_for_book_now": ""
    },
    {
        "intent_id": "fce813b37d2c111cd077f768409d60f1f61e08a5de51689ed1d9bf31b1d0e653",
        "transaction_number": "EWETU2ENA7EC",
        "orginal_amount": 1.5,
        "orginal_tip_amount": 0,
        "last_updated_tip_amount": 0,
        "updated_amount": 0.2,
        "transaction_date": "07-31-2022",
        "status": "Approved",
        "delivery_fee": 0,
        "time_left_for_approval": "",
        "expiry_date_for_book_now": ""
    },
    {
        "intent_id": "23563063de19dd87ae025e3572c5e91f3d39c52a3663dde6389a12f1f85db4f6",
        "transaction_number": "WUFU3EBAGUPU",
        "orginal_amount": 1.2,
        "orginal_tip_amount": 2.8,
        "last_updated_tip_amount": 2.8,
        "updated_amount": 2.1,
        "transaction_date": "07-31-2022",
        "status": "Approved",
        "delivery_fee": 0,
        "time_left_for_approval": "",
        "expiry_date_for_book_now": ""
    },
    {
        "intent_id": "e28daeacc94f26f3cc49344b2687bc7eb13188e448316bd290691e65fba83be0",
        "transaction_number": "JI8I1U1EWI0E",
        "orginal_amount": 2,
        "orginal_tip_amount": 0,
        "last_updated_tip_amount": 0,
        "updated_amount": 2,
        "transaction_date": "07-25-2022",
        "status": "Approved",
        "delivery_fee": 0,
        "time_left_for_approval": "",
        "expiry_date_for_book_now": ""
    },
    {
        "intent_id": "89dc62b3d2e78808c900431254803d674c1422ac9672b38a22ecc22bd0c17fda",
        "transaction_number": "ZU1UFUJAYIZU",
        "orginal_amount": 1,
        "orginal_tip_amount": 0,
        "last_updated_tip_amount": 0,
        "updated_amount": 1,
        "transaction_date": "07-12-2022",
        "status": "Approved",
        "delivery_fee": 0,
        "time_left_for_approval": "",
        "expiry_date_for_book_now": ""
    },
    {
            "intent_id": "b3fcc71955115fcc4f7622407baf3dc3ae225b173e9bc0daa83cb78b3f3d31c6",
            "transaction_number": "DI2EQEHABEVA",
            "orginal_amount": 1,
            "orginal_tip_amount": 0,
            "last_updated_tip_amount": 0,
            "updated_amount": 0.3,
            "transaction_date": "07-19-2022",
            "status": "Approved",
            "Delivery_fee":  0,
            "time_left_for_approval": "",
            "expiry_date_for_book_now": ""
          }
        ],
        "first_page_url": "http://canpay-api.local/integrator/loadmoretransactions?page=1",
        "from": 1,
        "last_page": 1,
        "last_page_url": "http://canpay-api.local/integrator/loadmoretransactions?page=1",
        "next_page_url": null,
        "path": "http://canpay-api.local/integrator/loadmoretransactions",
        "per_page": 10,
        "prev_page_url": null,
        "to": 9,
        "total": 9
    }
}
```

Note: The Accept Payment button in your dashboard should appear only if the expiry_date_for_book_now is not null. If it is null that means that the transaction has already been marked as payment accepted. Also, if the store is not admin driven, the Accept Payment button will not appear in the dashboard.The time_left_for_approval is the time left for consumer approval.

On Error:

```
{
  "code": 599,
  "message": "Invalid Intent ID.",
  "data": null
}
```

Call REST API for Dashboard Advanced search

To make an advance search in the dashboard for transaction data, the Merchant/Integrator needs to make a call to the CanPay server from the application backend.

API Sample

```
curl --location --request POST
'https://sandbox-api.canpaydebit.com/integrator/advancesearchtransactions' \
--form
'intent_id="9981dcba7ab19c1c37deae3e792c2b5e8f4cc0b49bdd43358ba994e2f94568ed"'
\
--form 'from_date="2022-09-15"' \
--form 'to_date="2022-09-20"' \
--form 'limit="10"' \
--form 'page="1"' \
--form 'transaction_number="1E1EZUFI8UCE"' \
--form 'amount=""' \
--form 'updated_amount="2"'
```

Parameters

- **Intent_id (Required):** Merchant/Integrator can find this id in the response from the authenticate API with the **auth_only** value set to **true**.
- **from_date (Optional) :** This should be the start date of the search criteria.
- **to_date (Optional):** This should be the end date of the search criteria.
- **limit** : Number of items you wish to display per page
- **page**: Use to get next/previous page items. The page value Initial 1. To get next page items then increases by 1.To get previous page items then decreases by 1.
- **transaction_number**: Merchant/Integrator can find this number in the response from the javascript processed_callback function.
- **amount**: Merchant/Integrator can find this number in the response from the javascript processed_callback function.

- **updated_amount**: The amount Merchant/Integrator has modified for the transaction. Also this can be found in get details of a payment/transaction API.

Example: If you have 30 records and you set the limit to 10. Now page 1 will return 10 records starting from 1. To get the next 10 records starting from 11 to 20 you should send the limit value as 10 and page value as 2. Don't change the limit value for next pages as it will lead to data mismatch.

Note: If you skip the from_date and to_date parameter the system will return the last 7 days data. Also, don't send the same date in from_date and to_date. To_date must be greater than from_date.

Response

On Success:

```
{
  "code": 200,
  "message": "All transactions fetched successfully.",
  "data": {
    "current_page": 1,
    "data": [
      {
        "intent_id": "012d364caa8412e84d3e42cf950d121e16e12c6082e1d56050e3d23f6d2651be",
        "transaction_number": "E7EVE5UDUYIV",
        "orginal_amount": 1,
        "orginal_tip_amount": 0,
        "last_updated_tip_amount": 0,
        "updated_amount": 2,
        "transaction_date": "09-19-2022",
        "status": "Approved",
        "delivery_fee": 0,
        "time_left_for_approval": "",
        "expiry_date_for_book_now": "09-26-2022"
      }
    ],
    "first_page_url": "https://sandbox-api.canpaydebit.com/integrator/advancesearchtransactions?page=1",
    "from": 1,
    "last_page": 1,
    "last_page_url": "https://sandbox-api.canpaydebit.com/integrator/advancesearchtransactions?page=1",
    "links": [
      {
        "url": null,
        "label": "pagination.previous",
        "active": false
      },
      {
        "url": "https://sandbox-api.canpaydebit.com/integrator/advancesearchtransactions?page=1",
        "label": "1",
        "active": true
      },
      {
        "url": null,
        "label": "pagination.next",
        "active": false
      }
    ],
    "next_page_url": null,
    "path": "https://sandbox-api.canpaydebit.com/integrator/advancesearchtransactions",
    "per_page": 10,
    "prev_page_url": null,
    "to": 1,
    "total": 1
  }
}
```

On Error:

```
{
  "code": 599,
  "message": "Invalid Intent ID.",
  "data": null
}
```

Call REST API to Accept Payment for Bank Posting

To mark a transaction as ready to post to the bank, the Merchant/Integrator needs to make a call to the CanPay server from the application backend. CanPay will mark these transactions as final and go to post to the bank for settlement.
API Sample

```
curl --location --request POST
'https://sandbox-api.canpaydebit.com/integrator/acceptpayment' \
--form
'intent_id="09ff8ec3717cc9b6f7e9a95a34be68547fb98d50157d268c49acfea82cdd9cc7"'
```

Parameters

- **Intent_id (Required):** Intent id is returned as a response after executing the Pay Via CanPay API from the widget.

Response

On Success:

```
{
  "code": 200,
  "message": "Payment acceptance marked successfully.",
  "data": null
}
```

On error

```
{
  "code": 599,
  "message": "No transaction found on your given input.",
  "data": null
}
```

```
{
  "code": 599,
  "message": "Transaction already marked as payment accepted.",
  "data": null
}
```

Note: This API endpoint was formerly known as https://sandbox-api.canpaydebit.com/integrator/booknow. All existing implementations with this endpoint are also going to return the same values. However it's recommended to use the new endpoint given above.

Call REST API to Add/Edit Plan Data

To add/edit plan data, the Merchant/Integrator needs to make a call to the CanPay server from the application backend.

API Sample

```
curl --location --request POST
'https://sandbox-api.canpaydebit.com/integrator/add-edit-plan' \
--form
'intent_id="0f47408fa9683ca6af26046fa4de2dc5bce56275586af23a6d679f5a59e25e0e"' \
--form 'plan_identifier="basic01"' \
--form 'plan_name="Basic Plan"' \
--form 'plan_details="Plan details"' \
--form 'amount="5"' \
--form 'status="active"' \
--form 'retain_old_plan_for_existing_subscribers="1"' \
--form
'plan_logo_url="https://sandbox-consumer.canpaydebit.com/img/canpay-logo.99cbd458.pn
g"'
```

Parameters

- **intent_id (Required):** Merchant/Integrator can find this id in the response from the authenticate API with the **auth_only** value set to **true**.
- **plan_identifier (Required):** Plan Identifier.
- **plan_name (Not Required when Integrator deactivates plan):** Plan Name.

- **plan_details (Not Required when Integrator deactivates plan):** Plan Description.
- **amount(Not Required when Integrator deactivates all plan w.f.t plan_identifier):** Plan Price.
- **status(Required):** Active or Deactive.
- **retain_old_plan_for_existing_subscribers(Optional):** Retain the old plan for specific/subscribed consumers.
- **plan_logo_url(Not Required when Integrator deactivates plan):** Gives exact plan logo URL.

Note: When Merchant/Integrator calls this API using plan_identifier(basic01). If basic01 is not already added to the integrator, it will be added, else it will be updated w.r.t integrator .

Response

On Success:

```
{
    "code": 200,
    "message": "Plan created successfully.",
    "data": null
}
```

```
{
    "code": 200,
    "message": "Plan updated successfully.",
    "data": null
}
```

On error

```
{
    "code": 599,
    "message": "Invalid Intent ID.",
    "data": null
}
```

Call REST API to Modify the Subscription

To modify a consumer subscription, the Merchant/Integrator needs to make a call to the CanPay server from the application backend.

API Sample

```
curl --location --request POST
'https://sandbox-api.canpaydebit.com/integrator/subscriptionupdate' \
--form
'intent_id="0f47408fa9683ca6af26046fa4de2dc5bce56275586af23a6d679f5a59e25e0e"' \
--form 'updated_plan_identifier="pro01"' \
--form 'consumer_identifier="70352e68e57843fbb3be74b75ff43521"' \
--form 'status="upgraded"' \
--form 'next_chrage_date="2022-11-30"' \
--form 'shipping_address="Haven Road Oliver Springs, TN 37840"'
```

Parameters

- **intent_id (Required):** Merchant/Integrator can find this id in the response from the authenticate API with the **auth_only** value set to **true**.
- **updated_plan_identifier (Not Required if the status is canceled):** Requested plan you want to update.
- **status(Required):** upgraded/downgraded/canceled.
- **next_chrage_date(Not Required if the status is canceled, Optional):** Next bill date.
- **shipping_address(Not Required if the status is canceled):** Product shipping address.
- **consumer_identifier (Required):** Merchant/Integrator can find this id returned as a response after executing the Subscribe Via CanPay API from the widgetResponse.

Note: If updated_plan_identifier does not exist in Canpay, then an error will show(Plan not found). At this time Merchant/Integrator needs to add a plan using Call REST API to Add/Edit Plan Data this API.

Note: The current plan is basic01 and the next bill date is 2022-11-25. If Merchant/Integrator calls this API using next_chrage_date = 2022-11-30 then the next bill date will be changed to 2022-11-30.

Response

On Success:

```
{
    "code": 200,
    "message": "Consumer plan upgraded successfully.",
    "data": null
}
```

```
{
    "code": 200,
    "message": "Consumer plan downgraded successfully.",
    "data": null
}
```

```
{
    "code": 200,
    "message": "Consumer plan canceled successfully.",
    "data": null
}
```

On error

```
{
    "code": 599,
    "message": "Invalid Intent ID.",
    "data": null
}
```

```
{
    "code": 599,
    "message": "Plan not found.",
    "data": null
}
```

Call REST API to Shipping Status Update

To update a consumer shipping status for every subscription history, the Merchant/Integrator needs to make a call to the CanPay server from the application backend.

API Sample

```
curl --location --request POST
'https://sandbox-api.canpaydebit.com/integrator/shipping-status-update' \
--form
'intent_id="0f47408fa9683ca6af26046fa4de2dc5bce56275586af23a6d679f5a59e25e0e"' \
--form 'plan_identifier="basic01"' \
--form 'consumer_identifier="70352e68e57843fbb3be74b75ff43521"' \
--form 'billing_date="2022-11-11"' \
--form 'status="fulfilled"'
```

Parameters

- **intent_id (Required):** Merchant/Integrator can find this id in the response from the authenticate API with the **auth_only** value set to **true**.
- **plan_identifier (Required):** Requested plan you want to update the shipping status.
- **consumer_identifier (Required):** Merchant/Integrator can find this id returned as a response after executing the Subscribe Via CanPay API from the widgetResponse.
- **billing_date(Required):** Bill date for finding particular subscription order history.
- **status(Required):** fulfilled.

On Success:

```
{
    "code": 200,
    "message": "Shipping status updated successfully.",
    "data": null
}
```

On error

```
{
    "code": 599,
    "message": "Invalid Intent ID.",
    "data": null
}
```

```
{
    "code": 599,
    "message": "Consumer subscription not found.",
    "data": null
}
```

Webhook Reference

CanPay will hit an API located on merchant/integrator’s end so that, if the network connection failed during the transaction process and Merchant/Integrator did not receive a response, CanPay will notify Merchant/Integrator about the transaction status later.

CanPay will also call this webhook whenever a modified transaction gets accepted or rejected by the consumer.

Merchant/Integrator needs to provide a POST method endpoint to make the call.

```
$ curl --location --request POST '<URL provided by Merchant/Integrator>' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--data-raw '{
   "intent_id": "f5622542dc8c32e9fcb29c50e618a9d3f2180e8b605314ea5618977c8ebefa6a",
   "amount": "15",
   “status”: “Success”,
   “expiry_date_for_book_now”: “”,
   "passthrough_param": null
}'
```

- **intent_id**: This intent_id is the id Merchant/Integrator received in response to the authorization API before the transaction.
- **transaction_number**: This is the CanPay transaction number sent to the Merchant/Integrator in response after the successful transaction.
- **amount**: Transaction amount Merchant/Integrator provided during the authorization API call and during the Configure the parameters for initialization.
- **status**: Current status of the transaction. It could be either Success or Failed.
- **expiry_date_for_book_now:** This is the expiry date for payment to be accepted. If it is null that means that the transaction has already been marked as payment accepted.
- **passthrough_param**: This is the Optional JSON parameter Merchant/Integrator provided during the Configure the parameters for initialization for reference.

Note: The system will call this API a maximum of 3 times per transaction in case the calling endpoint fails. The system will determine the success and the failed status from the CURL

response. If anything starts with 2 from the CURL response then the system will consider it as Success and anything that starts with other than 2 will be considered as Failed.

Endpoints: Sandbox vs Live

End Points	Sandbox	Live
To get intent_id	https://sandbox-api.canpaydebit.com/integrator/authorize	https://api.canpaydebit.com/integrator/authorize
Details of a payment/ transaction	https://sandbox-api.canpaydebit.com/integrator/transactiondetails	https://api.canpaydebit.com/integrator/transactiondetails
Void a payment/ transaction	https://sandbox-api.canpaydebit.com/integrator/transactionreversal	https://api.canpaydebit.com/integrator/transactionreversal
Get transaction modification reasons	https://sandbox-api.canpaydebit.com/integrator/getmodificationreason	https://api.canpaydebit.com/integrator/getmodificationreason
Modify transaction	https://sandbox-api.canpaydebit.com/integrator/transactionupdate	https://api.canpaydebit.com/integrator/transactionupdate
History and current status of modified transaction	https://sandbox-api.canpaydebit.com/integrator/modificationhistory	https://api.canpaydebit.com/integrator/modificationhistory
Data for e-Commerce Dashboard	https://sandbox-api.canpaydebit.com/integrator/dashboard	https://api.canpaydebit.com/integrator/dashboard
Load more data in e-Commerce Dashboard	https://sandbox-api.canpaydebit.com/integrat	https://api.canpaydebit.com/integrator/loadmoretransactions

	or/loadmoretransactions	
Dashboard Advanced search	https://sandbox-api.canpaydebit.com/integrator/advancesearchtransactions	https://api.canpaydebit.com/integrator/advancesearchtransactions
Accept Payment for Bank Posting	https://sandbox-api.canpaydebit.com/integrator/acceptpayment	https://api.canpaydebit.com/integrator/acceptpayment
Add/Edit Plan Data	https://sandbox-api.canpaydebit.com/integrator/add-edit-plan	https://api.canpaydebit.com/integrator/add-edit-plan
Modify the Subscription	https://sandbox-api.canpaydebit.com/integrator/subscriptionupdate	https://api.canpaydebit.com/integrator/subscriptionupdate
Shipping Status Update	https://sandbox-api.canpaydebit.com/integrator/shipping-status-update	https://sandbox-api.canpaydebit.com/integrator/shipping-status-update
Javascript File To launch the Widget with the intent_id	https://sandbox-remotepay.canpaydebit.com/cp-min.js	https://remotepay.canpaydebit.com/cp-min.js

HMAC Code Sample

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Security.Cryptography;
using System.Text;

namespace HMAC
{
      public class Program
      {
            public static void Main(string[] args)
            {
                  /* data to hash */
var data = "JSON_STRING_FROM_RESPONSE";
/* secret key */
var key = "API_SECRET";

            HMACSHA256 hashObject = new HMACSHA256(Encoding.UTF8.GetBytes(key));
            var signature =
hashObject.ComputeHash(Encoding.UTF8.GetBytes(data));
            var output=BitConverter.ToString(signature).Replace("-",
"").ToLower();
            /* print the output */
            Console.WriteLine(output);
            }
      }
}
```

Go

```go
package main

import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/hex"
    "fmt"
)

func main() {

/* data to hash */
 data := "JSON_STRING_FROM_RESPONSE"
/* secret key */
key := "API_SECRET"

    // Create a new HMAC by defining the hash type and the key (as byte
array)
    h := hmac.New(sha256.New, []byte(key))

    // Write Data to it
    h.Write([]byte(data))

    // Get result and encode as hexadecimal string
    khash := hex.EncodeToString(h.Sum(nil))
// print the output
    fmt.Println(khash)
}
```

Java

```java
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
```

```
public class HMAC {
      public static void main(String[] args) throws InvalidKeyException,
NoSuchAlgorithmException {

      /* data to hash */
      String data = "JSON_STRING_FROM_RESPONSE";
      /* secret key */
      String key = "API_SECRET";

      String khash = hmacWithJava("HmacSHA256", data, key);
      /* print the output */
            System.out.println(khash);
      }

      public static String hmacWithJava(String algorithm, String data,
String key)
                  throws NoSuchAlgorithmException, InvalidKeyException {
            SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(),
algorithm);
            Mac mac = Mac.getInstance(algorithm);
            mac.init(secretKeySpec);
            return bytesToHex(mac.doFinal(data.getBytes()));
      }

      private static final char[] HEX_ARRAY =
"0123456789ABCDEF".toCharArray();

      public static String bytesToHex(byte[] bytes) {
            char[] hexChars = new char[bytes.length * 2];
            for (int j = 0; j < bytes.length; j++) {
                  int v = bytes[j] & 0xFF;
                  hexChars[j * 2] = HEX_ARRAY[v >>> 4];
                  hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
            }
            return new String(hexChars);
      }
}
```

NodeJS

```
var crypto = require('crypto');

      /* data to hash */
var data = "JSON_STRING_FROM_RESPONSE";
/* secret key */
var key = "API_SECRET";

var khash = crypto.createHmac('SHA256', key).update(data).digest('hex');
/* print the output */
console.log(khash)
```

PHP

```
<?php
/* data to hash */
$data = "JSON_STRING_FROM_RESPONSE";
/* secret key */
$key = "API_SECRET"

$khash = hash_hmac("sha256",$data, $key);
/* print the output */
echo $khash;
?>
```

Python 2

```
import hmac
import hashlib
import base64

# data to hash
data = "JSON_STRING_FROM_RESPONSE"
# secret key
key = "API_SECRET"
hmac = hmac.new( key, data, hashlib.sha256 )
```

```
# print the output
print( hmac.hexdigest())
```

Python 3

```
import hmac, hashlib

data = 'JSON_STRING_FROM_RESPONSE' # Data to hash
key = 'API_SECRET' # Secret Key
key_bytes= bytes(key , 'latin-1') # Commonly 'latin-1' or 'utf-8'
data_bytes = bytes(data, 'latin-1') # Assumes `data` is also a string.
khash = hmac.new(key_bytes, data_bytes , hashlib.sha256).hexdigest()

# Print the output
print(khash)
```

Ruby

```
require 'openssl'

# data to hash
data = 'JSON_STRING_FROM_RESPONSE'

# secret key
key = 'API_SECRET'

khash = OpenSSL::HMAC.hexdigest(OpenSSL::Digest.new('sha256'), key, data)

# print the output
print khash
```

END OF DOCUMENT