REMITA COLLECTION INTEGRATION DOCUMENTATION

Component Name	Remita Collection Integration Documentation
General Description	Outlines Remita’s Collection API Integration methods
Target Audience	Integration Partners

Table of Contents

SECTION 1 OVERVIEW

Remita is a multi-bank, modular e-Payments, e-Invoicing, e-Collections, e-Payroll, and e-Schedules delivery solution available on mobile (as an app downloadable to your phone/device from https://play.google.com/store/apps/details?id=com.systemspecs.remitamobile&hl=en) and web (www.remita.net) platforms. It remits and collects funds to/from accounts in any Financial Institution via various payment channels (e.g. Bank Branch, Credit/Debit Cards, Internet Banking, Mobile Wallets, etc.).

Remita's Collection service enables a biller receive funds from customers/users via their website/web application. The customer/user visits the biller's website/web application to pass/send payment instruction to Remita. This means the customer/user selects a payment channel and the supplies required information on Remita's biller page. Payment is processed and customer/user account is debited while biller's account is credited.

This document describes Remita's Collection application programming interface (API) through which third party billers can integrate to the platform to use its various features via the accompanying exposed methods in the API. These, amongst others, include:

1. Generating RRR (Remita Retrieval Reference).
2. Making RRR payments.
3. Checking transaction status.

Full implementation of the API methods above will enable your payers visit your online portal to generate and pay Collection RRR's in your favour. You, the biller, will in turn be able to issue instructions on successful payments to the customer based on the value of services/products enjoyed from you within the limits agreed.

The following segments outline the details of our Collection API methods.

SECTION 2 Generating RRR

The first step to processing Collection on Remita is generating a Remita Retrieval Reference (RRR). This is typically done by a payer. If you have implemented the method below per specifications outlined, your customers will be able to visit your online portal to generate a Collection RRR.

There are four ways to generate an RRR;

1. Single payment RRR
2. Single payment RRR with custom fields
3. Split payment RRR
4. Split payment RRR with custom fields.

1. **Single payment RRR:** this is implemented when the total amount collected from your customer is expected to go into one collection account.

2. **Single payment RRR with custom fields:** This is similar to the RRR generation above. The only difference is custom fields which is sent along with other payment details. Custom fields are your unique fields.

3. **Split payment RRR:** this is implemented when the total amount collected from your customer is expected to be split into several collection accounts or merchants.

4. **Split payment RRR with custom fields**: This is similar to the RRR generation above. The only difference is custom fields which is sent along with other payment details. Custom fields are your unique fields.

2.1 Generate RRR

Fields and Values

Parameter Name	Description	Type
merchantId	Required This is a unique identifier for the Biller	String

serviceTypeId	Required This is a unique identifier for service type receiving the payment	String
orderId	Required This is the Biller Transaction ID	String
Authorization	Required remitaConsumerKey={merchantID},remitaConsumerToken={SHA512(merchantId+ serviceTypeId+ orderId+totalAmount+apiKey)}	String
payerName	Required This is the name of the customer to be displayed on the payment page.	String
payerEmail	Required This is the Payer's Email Address	String
payerPhone	Optional This is the Payer's Phone Number	Numeric
amt	Required This is the total monetary value of the transaction.	Numeric
description	Required Details of the service your customer is paying for.	String
customFields	**Optional (Used only when customers have custom fields)** Name : Name of the file Value: Value of the field Type: variable type (e.g String, Integer, float). Specify "ALL" for Remita to accept all variable types.	
lineItems	**Optional (Used only when the amount is to be split between different accounts)** lineItemsId – Unique identifier for the line items beneficiaryName – Name of the account beneficiaryAccount – Account number bankCode- CBN bank code to identify each banks beneficiaryAmount – A percentage of the total amount for the account deductFeeFrom – Specifies the line item Remita is to deduct her total transaction fee for all line items. Either 0 or 1.	

2.2 Sample Code

METHOD: POST

HEADER
Authorization: remitaConsumerKey={merchantID},remitaConsumerToken={SHA512(merchantId+ serviceTypeId+ orderId+totalAmount+apiKey)}

URL: http://www.remitademo.net/remita/exapp/api/v1/send/api/echannelsvc/merchant/api/paymentinit

Sample request for single payment.

```
{
         "serviceTypeId": "4430731",
         "amount": "20000",
         "orderId": "221028",
        "payerName": "Ogunseye Olanrewaju",
        "payerEmail": "awoedey2k@gmail.com",
        "payerPhone": "2347038496242",
         "description": "Payment for Donation 3",
}
```

Sample request for single payment with custom fields.

```
{
         "serviceTypeId": "4430731",
         "amount": "20000",
         "orderId": "221028",
        "payerName": "Ogunseye Olanrewaju",
        "payerEmail": "awoedey2k@gmail.com",
        "payerPhone": "2347038496242",
         "description": "Payment for Donation 3",
         "customFields": [{
                  "name": "Matric Number",
                  "value": "1509329285795",
                  "type": "ALL"
         },
        {
                  "name": "Invoice Number",
                  "value": "1234",
                  "type": "ALL"
         }]
}
```

Sample request for split payment.

```
{
            "serviceTypeId": "4430731",
            "amount": "20000",
            "orderId": "221028",
           "payerName": "Ogunseye Olanrewaju",
           "payerEmail": "awoedey2k@gmail.com",
           "payerPhone": "2347038496242",
            "description": "Payment for Donation 3",
           "lineItems":[{
                       "lineItemsId":"itemid1","beneficiaryName":"Alozie Michael",
              "beneficiaryAccount":"0360883515","bankCode":"020","beneficiaryAmount":"7000","deductFeeFrom":"1"
            },
            {
                       "lineItemsId":"itemid2","beneficiaryName":"Folivi Joshua",
              "beneficiaryAccount":"4017904612","bankCode":"022","beneficiaryAmount":"3000","deductFeeFrom":"0"
            }]
}
```

Sample request for split payment with custom fields.

```
{
            "serviceTypeId": "4430731",
            "amount": "20000",
            "orderId": "221028",
           "payerName": "Ogunseye Olanrewaju",
           "payerEmail": "awoedey2k@gmail.com",
           "payerPhone": "2347038496242",
            "description": "Payment for Donation 3",
           "customFields": [{
                       "name": "Matric Number",
                       "value": "1509329285795",
                       "type": "ALL"
            },
           {
                       "name": "Invoice Number",
                       "value": "1234",
                       "type": "ALL"
            }]
           "lineItems":[{
                       "lineItemsId":"itemid1","beneficiaryName":"Alozie Michael",
              "beneficiaryAccount":"0360883515","bankCode":"020","beneficiaryAmount":"7000","deductFeeFrom":"1"
            },
            {
                       "lineItemsId":"itemid2","beneficiaryName":"Folivi Joshua",
              "beneficiaryAccount":"4017904612","bankCode":"022","beneficiaryAmount":"3000","deductFeeFrom":"0"
            }]
}
```

Sample response

```
jsonp ({"statuscode":"025","RRR":"260007663696","status":"Payment Reference generated"})
```

SECTION 3 Initiating RRR payments

After generating Remita Retrieval Reference (RRR) via your online portal successfully, your payers have to make payment using the RRR. Your customers need to ensure that the designated paying accounts are always funded to cover the transactions accordingly.

To make payment for Remita Retrieval Reference (RRR), a HTTP POST FORM should be sent to Remita.

3.1 Sample RRR Payments HTML Form

Fields and Values

Parameter Name	Description	Type
merchantId	**Required** This is a unique identifier for the Biller	String
rrr	**Required** This is the Remita Retrieval Reference	String
hash	**Required** merchantId+rrr+api_key	String
responseurl	**Required** The URL to which Remita should send transaction status report to on completion of transaction.	String

Generating Request Hash

For security reasons you are required to hash your payment details with your API Key. Upon registration on Remita you will be given an API Key which should be kept secret. A valid payment request hash is generated by concatenating the following payment details and hashed using SHA512 algorithm and the assigned API Key: merchantId+rrr+api_key

Sample RRR Payments HTML Form

```
<html>
    <body>
        <form action="http://www.remitademo.net/remita/ecomm/finalize.reg" name="SubmitRemitaForm" method="POST">
            <input name="merchantId" value="1509328648353" type="hidden">
             <input name="hash" value="ABCED12D3E1476DEFA12" type="hidden">
            <input name="rrr" value="Y11095959" type="hidden">
            <input name="responseurl" value="http://www.yourwebsite.com/response.php" type="hidden">
            <input type ="submit" name="submit_btn" value="Pay Via Remita">
        </form>
    </body>
</html>
```

SECTION 4 Transaction Status

You can query the status of a transaction after payment transaction has been initiated. The status of the transaction is determined using the status code/status message. For successful payments, status code is either “00” or “01”.

4.1 Status Check Parameter

Generating Request Hash
For security reasons you are required to hash your payment details with your API Key. Upon registration on Remita you will be given an API Key which should be kept secret. A valid payment request hash is generated by concatenating the following payment details and hashed using SHA512 algorithm and the assigned API Key: RRR/OrderID+api_key+merchantId

Fields and Values

Parameter Name	Description	Type
merchantId	Required This uniquely identifies the biller.	String
rrr	Required The Remita Retrieval Reference	String
hash	Required SHA512 (RRR/OrderID+api_key+merchantId)	String
OrderID	Required This is the billers transaction ID	String

Payment Status URL using RRR

Request Method: GET

WEB URL: http://www.remitademo.net/remita/ecomm/{merchantId}/{RRR}/{hash}/status.reg

Payment Status URL using OrderID

Request Method: GET

WEB URL:
http://www.remitademo.net/remita/ecomm/{merchantId}/{OrderID}/{hash}/orderstatus.reg

Sample JSON Response

```
{
   "statusmessage": "Transaction Approved",
   "merchantId": "1509328648353",
   "status": "01",
   "RRR": "O11615747",
   "transactiontime": "2014-08-04 01:39:48 PM",
   "orderId": "1021232"
}
```

SECTION 5 Payment Notification URL

This is a listening endpoint that receives transaction updates via a JSON post from Remita for transactions processed via channels such as Bank Branch, POS and Internet Banking while feedback for transactions processed via debit cards is returned to your system via a redirect to your response URL. When your payment notification listener receives the notification from Remita, you are expected to make a get status call using the orderId/orderRef via the transaction status API which can be found in section 4 of this document.

5.1 Sample JSON payload sent to your payment Notification URL

```
[
 {
   "rrr":"130007649273",
   "channel":"BRANCH",
   "amount":7500.00,
   "transactiondate":"03/10/2017",
   "debitdate":"03/10/2017",
   "bank":"011",
   "branch":"011152387",
   "serviceTypeId":"504940131",
   "dateRequested":"03/10/2017",
   "orderRef":"031017110736",
   "payerName":"alozie michael",
   "payerPhoneNumber":"08146963838",
   "payerEmail":"alozienedu@gmail.com",
   "uniqueIdentifier":""
  }
]
```

NB: Kindy check the status of the RRR; if status code is 00 or 01, kindly update status of the transaction on your database and return **"OK"**. Else return **"not ok"**.

SECTION 6 APPENDIX

6.1 Transaction Codes

Code	Description
00	Transaction Completed Successfully
01	Transaction Approved
02	Transaction Failed
012	User Aborted Transaction
020	Invalid User Authentication
021	Transaction Pending
022	Invalid Request
023	Service Type or Merchant Does not Exist
025	Payment Reference Generated
029	Invalid Bank Code
030	Insufficient Balance
031	No Funding Account
032	Invalid Date Format
040	Initial Request OK
999	Unknown Error

6.2 Bank Codes

Bank	Code
ACCESS BANK PLC	044
CITI BANK	023
DIAMOND BANK PLC	063
ECOBANK NIGERIA PLC	050
FIDELITY BANK PLC	070
FIRST BANK OF NIGERIA PLC	011
FIRST CITY MONUMENT BANK PLC	214
GUARANTY TRUST BANK PLC	058
HERITAGE BANK	030
JAIZ BANK	301
KEYSTONE BANK	082
SKYE BANK PLC	076
STANBIC-IBTC BANK PLC	039
STANDARD CHARTERED	068
STERLING BANK PLC	232
UNION BANK OF NIGERIA PLC	032
UNITED BANK FOR AFRICA PLC	033
UNITY BANK PLC	215
WEMA BANK PLC	035
ZENITH BANK PLC	057
CBN	000
PROVIDOUS	101
SUNTRUST	100