LiquidNet Send Transaction API Specifications

Version 1.7

Date: April 12th, 2024

Revision History

Version	Prepared By	Reviewed By	Date	Modifications Done
1.0	Sanad Nepal	Subina Manandhar	02-20-2021	- Reviewed and Baselined
1.1	Bishal Poudel	Aashish Adhikari	12-20-2021	- QueryTXNStatus API method updated from GET to POST - Section 1.2 updated with steps from hmac signature generation
1.2	Aashish Adhikari		12-28-2021	- New error codes added in Appendix 1 - Sections re-arranged - Callback feature information added in section 2.7.2
1.3	Aashish Adhikari		01-11-2022	- Removed deprecated catalogue type AGT from the GetCatalogue API - Request parameter update on CheckBalance API - Added new error codes - calcBy option 'C' removed in sendTransaction API. Use only calcBy as 'P' - Removed ADO which was a redundant catalogue type
1.4	Pratisha Shakya	Aashish Adhikari	03-21-2022	Additional fields added in SendTransaction API (section 2.5) - RemitterType, - BeneficiaryType, - SenderNativeAddress - ReceiverNativeAddress - SenderNativeMiddleName - ReceiverNativeMiddleName
1.5	Aashish Ghaju	Pratisha Shakya	10-18-2022	Reviewed and Updated - Mode of Payment - Additional fields in SendTransaction API - Transaction Status list from QueryTXNStatus

1.6	Aashish Ghaju	Aashish Adhikari	07-25-2023	- Added new endpoint: DocUpload - Updated Request URL - Deleted the API that Lightnet no longer provides: CancelTransaction, AmendTransaction, and ReconcileReport
1.7	Subista Shakya	Aashish Adhikari	04-12-2024	- Added new endpoints: (1) Rate History Daily, and (2) Get Branch - Added new payment modes in SendTransaction, GetFieldInfo, GetAgentList and GetEXRate API endpoints o Individual to Individual Bank Transfer(V) o Individual to Business Bank Transfer (X) o Business to Individual Bank Transfer (Y) o Business to Business Bank Transfer (Z) - Added new parameter “RemitCurrency” in the following endpoints for Multi-Currency API o GetCatalogue o GetAgentList o GetEXRate o Commit Txn o QueryTxnStatus o Reconcile Report o Cancel Txn o Amend Txn o GetFieldInfo - Removed Upload Document API

Contents

Contents

1 Introduction

This document contains the API specifications required for a Sending party to integrate RemitX API to be able to send transactions to allowed corridors.

The sections below provide an overview of the methods declared in the API, the API call flow, and the generation of signature for authentication

1.1 List of methods, declared in API

Method	Description
GetEcho	Test Connection
GetCatalogue	Get the Static Data • Countries that are available, • Payment Mode • Get Payout Agent List • Sender Occupation • Source of Fund • Relationship • Purpose of Remittance
GetAgentList	Get all the agent list and location ID
GetEXRate	Get Exchange Rate and calculate Service Fee
GetFieldInfo	Retrieve the Required Fields for Send Transaction
SendTransaction	Send Transaction
QueryTXNStatus	Check the status of TXN
ReconcileReport	Generate TXN Sent/Paid/Cancel Status Report
CommitTransaction	To Commit Transaction call CommitTransaction. This function has to be called within 30 min after SendTransaction.
DocUpload	Upload documents by document type
AmendTransaction	This method is used to amend the transaction which partner has posted
CancelTransaction	Partner can call this function to cancel transaction that is posted.
CheckBalance	To enquire Partner Balance
InquiryAccount	To check valid Beneficiary Account Number
GetTransactionDetail	To get the detail information of Transaction for Payout
ConfirmPayTransaction	To make Payout Transaction as PAID or Completed

1.2 *Process to send transaction:*

1) Call *GetCatalogue* to update the catalogue data set required to send a transaction.
2) Call *GetAgentList* to get all the active payout location details.
3) Call *InquiryAccount* to valid beneficiary account number if it is for Indonesia bank transfer.
4) Call *GetEXRate* method to get the latest exchange rate and service charge based on location ID.
5) Call *GetFieldInfo* function to retrieve the Required Fields for each Location ID which needs to be passed in the *SendTransaction* function
6) Call *SendTransaction* to send transactionwith the relevant parameters.
7) In Response to *SendTransaction* we provide you CONFIRMATIONID, which must be passed at the time of calling *CommitTransaction* to commit the transaction.
8) You must call this function within 30 minutes after *SendTransaction*. After Transaction is Authorized, it will be ready for payment.
9) Web Services acknowledge Partners request and notify the confirmed transaction status within 180 seconds.
10) Partners can use *QueryTXNStatus* method to check transaction status if network timeout or other error occurs.
11) Partners can initiate server warm-up or keep-alive with REST Method *GetEcho* from time to time

Partners must supply Technical team with a FIXED IP address to allow access into the web service applications

All our Methods return (Response) <CODE> parameter as 0 (Zero) which denotes query execution is SUCCESS. For more detail for <Code> please refer to Appendix.

1.3 How to create a Signature for HMAC Authentication?

First, the client needs to create a string (MAC – Message Authentication Code) which will contain all the request data that the client wants to send to the server. Generally, the string contains the following parameters

1. HTTP method (should be uppercase)

HTTP Method type for the request object.

2. APP Id and API Key

This will be provided from API Provider. The APP Id is the username of the API user and API Key is the password for the same setup through API Password Setup email link.

3. Nonce

The Nonce is a random number or string which is used only once per request.

4. The request URI

The request URI is a full encoded URI as

```
string requestUri = HttpUtility.UrlEncode(url.ToLower());
```

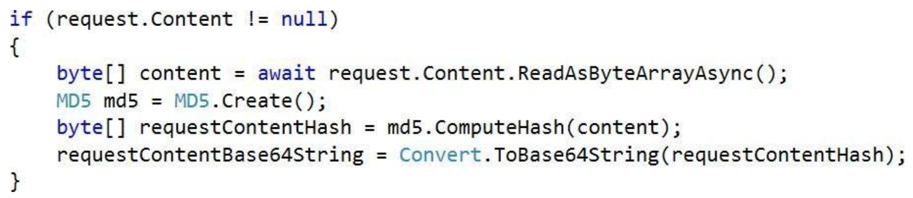
5. Request timestamp

We need to calculate the Request Time Stamp value by using the UNIX time (number of seconds since Jan. 1st 1970). We need to do this to overcome the possibility of different time zone issues between the client and the server.

6. Base 64 string representation of the request payload (request body) If the

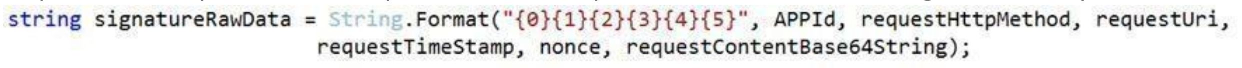
request contains a body then we need to do the following things.

First, we need to use MD5 hashing algorithm to hash the body content, then we need to convert that hash into a Base64 string. The following code does the same.



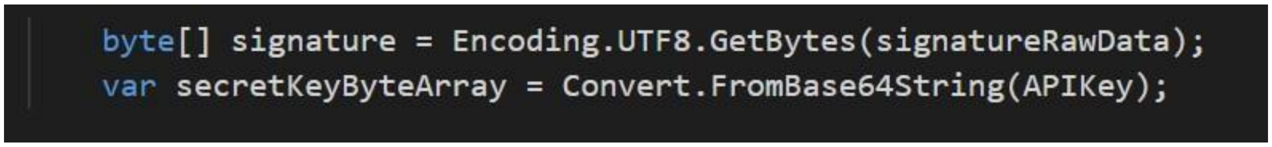
```
if (request.Content != null)
{
    byte[] content = await request.Content.ReadAsByteArrayAsync();
    MD5 md5 = MD5.Create();
    byte[] requestContentHash = md5.ComputeHash(content);
    requestContentBase64String = Convert.ToBase64String(requestContentHash);
}
```

Then, we need to build the signature raw data by combining APP Id, request HTTP Method Type, request URI, request timestamp, nonce, and request Content Base64 String without any delimiters.



```
string signatureRawData = String.Format("{0}{1}{2}{3}{4}{5}", APPId, requestHttpMethod, requestUri,
                          requestTimeStamp, nonce, requestContentBase64String);
```

And, we need to convert the signature raw data into a byte array and also the key to Base64 string.



```
byte[] signature = Encoding.UTF8.GetBytes(signatureRawData);
var secretKeyByteArray = Convert.FromBase64String(APIKey);
```

Finally, client needs to generate HMAC signature using the shared secret Base64 string API Key and set it in Authorization header in format shown below.

[Authorization: hmacauth APPId:Signature:Nonce:Timestamp]

7. The request URI

The request URI is a full encoded URI as

```
string requestUri = HttpUtility.UrlEncode(url.ToLower());
```

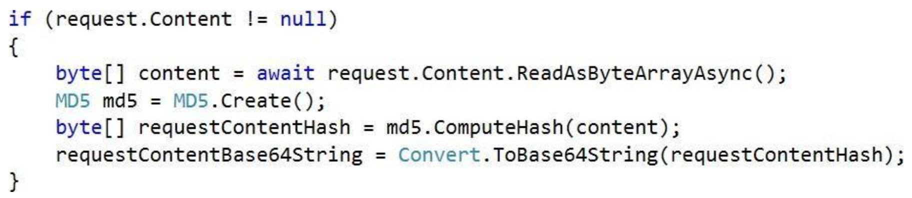
8. Request timestamp

We need to calculate the Request Time Stamp value by using the UNIX time (number of seconds since Jan. 1st 1970). We need to do this to overcome the possibility of different time zone issues between the client and the server.

9. Base 64 string representation of the request payload (request body) If

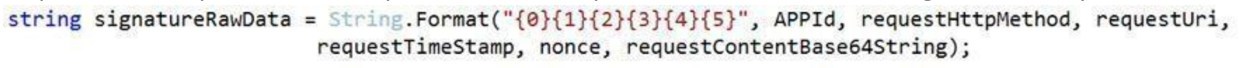
the request contains a body then we need to do the following things.

First, we need to use MD5 hashing algorithm to hash the body content, then we need to convert that hash into a Base64 string. The following code does the same.



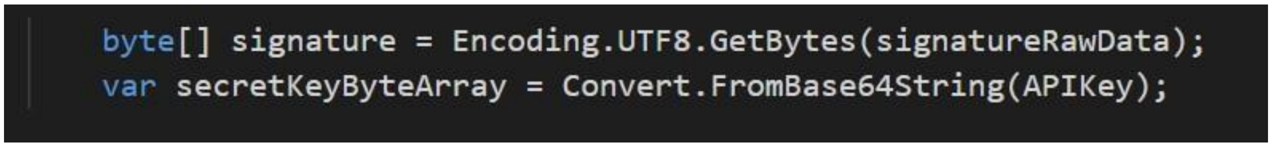
```
if (request.Content != null)
{
    byte[] content = await request.Content.ReadAsByteArrayAsync();
    MD5 md5 = MD5.Create();
    byte[] requestContentHash = md5.ComputeHash(content);
    requestContentBase64String = Convert.ToBase64String(requestContentHash);
}
```

Then, we need to build the signature raw data by combining APP Id, request HTTP Method Type, request URI, request timestamp, nonce, and request Content Base64 String without any delimiters.



```
string signatureRawData = String.Format("{0}{1}{2}{3}{4}{5}", APPId, requestHttpMethod, requestUri,
                          requestTimeStamp, nonce, requestContentBase64String);
```

And, we need to convert the signature raw data into a byte array and also the key to Base64 string.



```
byte[] signature = Encoding.UTF8.GetBytes(signatureRawData);
var secretKeyByteArray = Convert.FromBase64String(APIKey);
```

Finally, client needs to generate HMAC signature using the shared secret Base64 string API Key and set it in Authorization header in format shown below.

[Authorization: hmacauth APPId:Signature:Nonce:Timestamp]

Sample code in C# .NET is provided in

Communication Timeouts and Wait Periods

The timeout period for each unique transaction is 180 seconds. If a transaction request to web services is aborted or closed as part of network error detection and recovery before the timeout, Partner should query the transaction status using "*QueryTXNStatus*" before retrying the same transaction.

2 Methods and Specifications

This section illustrates the various methods required for integration and their specifications

2.1 GetEcho

To ensure optimal response time from the Web Service, Partner can invoke GetEcho method from time to time for the purpose of server 'warm-up' or keep-alive purpose.

Request URL

POST {connectbaseurl}/api/webservice/GetEcho

Request:

```
{
        "agentSessionId": "67898987979"
}
```

Response:

```
{
    "code": "0",
    "message": "Success."
}
```

Field	Description
code	0 means System is Up
message	If Error then Error Message is returned

2.2 GetCatalogue

Call this method to Get the Static Data like Countries, list Payment Mode, Get Payout Agent List

Get Available Country List

The List of all available Countries along with the ISO -3 Country Codes are returned in response, these Codes are to be passed in all Functions where Country Code is required.

webservice/

POST {connectbaseurl}/api/webservice/GetCatalogue

Request:

```
{
  "agentSessionId": "string",
  "catalogueType": "string",
  "additionalField1":
  "string",
  "additionalField2":
  "string",
  "additionalField3":
  "string",
  "remitCurrency": "string"
}
```

Field	Data Type	Max Length	Description	Required
agentSessonId	string	150	WS System will response same agent session id to track the response is valid from same session	Y
catalogueType	String	3	Catalogue type CTY: Get Available Country STA: Get Available States of available Countries PTY: Get Available Payment Mode OCC: Get Occupation List SOF: Get Source of Fund REL: Get Relationship list POR : Get Purpose of Remittance DOC : Get Customer Document ID Type ART: Get Area/Town/City ACT: Account Type	y
additionalField1	String	3	*Pass ISO -3 Country Code if Catalogue Type is PTY	*

additionalField2	String	50	*Pass returned states code from Catalogue ***STA*** If Catalogue is ***ART***	*
additionalField3	String	50	Future Use	
Remit Currency	String	3	ISO3 Send Amount Currency Code	* ***Mandatory if partner account is setup as multiple sending currency account***

Response:

Field	Description
code	If Error then Error Code is returned
agentSessonId	Partner Session ID
message	If Error then Error Message is returned
data	Catalogue Data / ID that you might need to pass to API if required
value	Catalogue Value

To get the Catalogue value you should pass as follows example:

CTY: Get Available Country

```
"catalogueType":"CTY"
"additionalField1":""
"additionalField2":""
"additionalField3":""
```

STA: Get Available States of Countries

```
"catalogueType":"STA"
  "additionalField1":"KOR"
  "additionalField2":""
  "additionalField3":""
```

ADDITIONALFIELD1: Destination/Originating ISO 3 Country Code

PTY: Get Available Payment method for PHILIPPINES country

Additional Field 1 should contain ISO -3 Country Code for Philippines

```
"catalogueType":"PTY"
  "additionalField1":"PHL"
  "additionalField2":""
  "additionalField3":""
```

ADDITIONALFIELD1: Destination country Name

OCC: Get Occupation List

```
"catalogueType":"OCC"
  "additionalField1":""
  "additionalField2":""
  "additionalField3":""
```

SOF: Get Source of Fund

```
"catalogueType":"SOF"
```

```
 "additionalField1":""
 "additionalField2":""
 "additionalField3":""
```

REL: Get Relationship list

```
"catalogueType":"REL"
 "additionalField1":""
 "additionalField2":""
 "additionalField3":""
```

POR : Get Purpose of Remittance

```
"catalogueType":"POR"
 "additionalField1":""
 "additionalField2":""
 "additionalField3":""
```

DOC : Get Customer Document ID Type

```
"catalogueType":"DOC"
"additionalField1":""
"additionalField2":"
"additionalField3":""
```

ART: Get Area/Town

```
"catalogueType":"ART"
 "additionalField1":""
 "additionalField2":""
 "additionalField3":""
```

ACT: Account type

```
"catalogueType":"ACT"
 "additionalField1":""
 "additionalField2":""
 "additionalField3":""
```

ADDITIONALFIELD1: Destination/Originating ISO 3 Country Code
ADDITIONALFIELD2: State Code return from Catalogue Type ***STA***

2.3 GetAgentList

Call *GetAgentList* to get the detailed location information of Payout Partners from System.Location ID should be passed to calculate *GetExRate* and to call *SendTransaction* method.

Request URL

POST {connectbaseurl}/api/webservice/GetAgentList

Request:

```
{
  "agentSessionId":"string",
  "paymentMode": "string",
  "payoutCountry": "string",
  "remitCurrency": "string"
}
```

Field	Data Type	Max Length	Description	Required
agentSessionId	string	150	WS System will response same agent session id to track the response is valid from same session	y
paymentMode	String	1	B- Account Deposit C- Cash Pickup P- Corporate Payment R- Card Payment W- Mobile Wallet V-Individual to Individual Bank Transfer X-Individual to Business Bank Transfer Y-Business to Individual Bank Transfer Z- Business to Business Bank Transfer *Refer **GetCatalogue** function CATALOGUETYPE: PTY*	y
payoutCountry	String	3	ISO -3 Country Code For Example: PHL for Philippines *Refer **GetCatalogue** function CATALOGUETYPE: CTY*	y
remitCurrency	String	3	ISO3 Send Amount Currency Code	**** Mandatory if partner account is setup as multiple sending currency account***

Response:

```
{
        "code": "0",
        "locationDetail": [
            {
                  "locationId": "1071",
                  "locationName": "AGRICULTURAL DEVELOPMENT BANK LTD.",
                  "optionalField": ""
            },
            {
                  "locationId": "1071",
                  "locationName": "BANK OF KATHMANDU LUMBINI",
                   "optionalField": ""
            },
            {
                  "locationId": "1071",
                  "locationName": "CITIZEN BANK INTERNATIONAL LTD",
                  "optionalField": ""
            },
            {
                  "locationId": "1071",
                  "locationName": "EVEREST BANK LIMITED",
                  "optionalField": ""
            },
      ]
}
```

Field	Description
code	0 if Success otherwise Error Code
locationId	LocationID is Payout Branch Location ID (Branch ID) or BANK ID, This value has to be passed while calling SendTransaction/GetExRate.
locationName	Name of Payout Location or Bank Name in the case of Account Deposit
optionalField	Reserved for future use

2.4 GetBranch

Partner call *GetBranch* to get the branch list for the corridors that requires Bank Branch Code.

Request URL:
POST {connectbaseurl}/api/webservice/getbranch

Request:

```
{
 "agentSessionId": "string",
"locationId": "string",
 "transferAmount": "string"
}
```

2.5 GetEXRate

Partner call *GetEXRate*Command to get the latest Exchange Rate and Service Charges.

Request URL

POST {connectbaseurl}/api/webservice/GetExRate

Request:

```
{
    "agentSessionId":"string",
    "transferAmount":"string",
    "calcBy": "string",
    "payoutCurrency":"string",
    "paymentMode": "string",
    "locationId": "string",
    "payoutCountry":"string",
    "promotionCode":"string",
    "remitCurrency": "string"
  }
```

Field	Data Type	Max Length	Description	Required

agentSessionId	string	150	Partner ID, WS System will response same agent session id to track the response is valid from same session	y
transferAmount	Money		Amount to be deposited/received by beneficiary/receiver in the payout country currency	y
calcBy	String	1	P – Calculate by Payout Currency	y
payoutCurrency	String	3	ISO3 Payout Amount Currency Code	y
paymentMode	String	1	B- Account Deposit C- Cash Pickup P- Corporate Payment R- Card Payment W- Mobile Wallet V-Individual to Individual Bank Transfer X-Individual to Business Bank Transfer Y-Business to Individual Bank Transfer Z- Business to Business Bank Transfer *Refer **GetCatalogue** function CATALOGUETYPE: PTY*	y
locationId	String	10	Payout Branch Location ID (Branch ID) or BANK ID Refer to *GetAgentList* method	y
payoutCountry	String	3	ISO -3 Country Code For Example: PHL for Philippines *Refer **GetCatalogue** function CATALOGUETYPE: CTY*	y
promotionCode	String	20	Based on the Promotion code provided, system return premium rate and discounted fee *Only if Promotion is activated and to be consumed.	*
remitCurrency	String	3	ISO3 Send Amount Currency Code	** Mandatory if partner account is setup as multiple sending currency account*

Response:

```
{
    "code": "string",
    "agentSessionId": "string",
    "message": "string",
    "collectAmount": "Money",
    "collectCurrency": "string",
    "serviceCharge": "Money",
    "gstCharge": "Money",
    "transferAmount": "Money",
    "exchangerate": "Float",
    "payoutAmount": "Money",
    "payoutCurrency": "string",
    "feeDiscount": "Money",
    "additionalPremiumRate": "Float",
    "settlementRate": "Float",
}
```

Field	Description
code	0 if Success otherwise Error Code
agentSessionId	Partner Session ID
message	Message to Partner, for example Payout Location detail
collectAmount	Remit Amount including service fee
collectCurrency	Collect ISO 3 Currency Code e.g. MYR, USD, KRW
serviceCharge	Service Charge in Collect Currency
gstCharge	GST Charge on Service Fee if applicable
transferAmount	Remit Amount excluding service fee
exchangerate	Exchange Rate as of date
payoutAmount	Total Payout Amount to Payout Location
payoutCurrency	Payout Currency
feeDiscount	If PROMOTIONCODE has Fee Discount, it returns discounted Fee Value
additionalPremiumRate	If Promotion Code has Premium Rate, it will return Additional Premium Rate
settlementRate	Partner Settlement Rate

LIGHTNET

2.5.1 Rate History Daily

Call *GetRateHistory* to view one day rate detailed of Payout Partners from System

Request URL

POST {connectbaseurl/api/WebService/getratehistory

Request:

```
{
  "agentSessionId":
  "string", "date":
  "string", "payoutCountry":
  "string",
  "payoutCurrency":
  "string",
}
```

Field	Data Type	Max Length	Description	Required
agentSessionId	string	150	WS System will response same agent session id to track the response is valid from same session	Y
date	string	10	Rate Date format: YYYY-MM-DD	Y
payoutCountry	string	3	ISO3 Payout Country Code	Y
payoutCurrency	string	3	ISO3 Payout Amount Currency Code	Y

Response:

```
{
        "code":”0”,
        "agentSessionId":”string”,
        "message": "string",
        "rateHistory": [
              {
                    "rate": "string",
                    "updatedDate": "string"
              }
        ]
}
```

Field	Description
code	0 if Success otherwise Error Code
agentSessionId	Partner Session ID
message	System Message
rate	Exchange rate as recorded
updatedDate	Updated rate of Date and Time as recorded

2.6 *SendTransaction*

After successfully calling the *GetExRate* method, call this function to send a transaction. While sending a Transaction if Customer already exists, it will update the existing Customer. If the Customer is new, the *SendTransaction* method *Auto-creates* new Customer based on SenderIDType and SenderIDNumber(these information will be unique to identify Customer)

The *CommitTransaction* method has to be called within 30 minutes of *SendTransaction.*

If Partner RESENDS the same AGENTTXNID, System will detect it as a *Duplicate* Transaction and it will not Store the subsequent Transaction. The possible Error Codes and Responses in case of Duplication would be:

<CODE>1</CODE>– The First attempted AGENTTXNID has been successfully accepted and Authorized as well. The Transaction will then be in Hold, Sanction or Compliance Status. *QueryTXNStatus* can be called to view real Status.

<CODE>2</CODE> – The First attempted AGENTTXNID has been accepted and is in Un-Commit-Hold or Un-Commit-Compliance Status. It requires authorization by calling *CommitTransaction* using the same CONFIRMATIONID from *SendTransaction* Response.

For diagrammatic process flow, refer to Appendix 2.

webservice/

POST {connectbaseurl}/api/webservice/SendTransaction

Request:

```
{
  "agentSessionId": "string",
  "agentTxnId": "string",
  "locationId": "string",
  "senderFirstName": "string",
  "senderMiddleName": "string",
  "senderLastName": "string",
  "senderGender": "string",
  "senderAddress": "string",
  "senderCity": "string",
  "senderState": "string",
  "senderZipCode": "string",
  "senderCountry": "string",
  "senderMobile": "string",
  "senderNationality": "string",
  "senderIdType": "string",
  "senderIdNumber": "string",
"senderIdIssueCountry": "string",
  "senderIdIssueDate": "string",
  "senderIdExpireDate": "string",
  "senderDateOfBirth": "string",
  "senderOccupation": "string",
  "senderSourceOfFund": "string",
  "senderSecondaryIdType": "string",
  "senderSecondaryIdNumber": "string",
  "senderEmail": "string",
  "senderNativeFirstname": "string",
  "senderNativeMiddlename": "string",
  "senderNativeLastname": "string",
  "senderBeneficiaryRelationship": "string",
  "purposeOfRemittance": "string",
  "receiverFirstName": "string",
  "receiverMiddleName": "string",
  "receiverLastName": "string",
  "receiverAddress": "string",
  "receiverDateOfBirth": "string",
  "receiverGender": "string",
  "receiverContactNumber": "string",
  "receiverState": "string",
  "receiverAreaTown": "string",
  "receiverCity": "string",
  "receiverZipCode": "string",
  "receiverCountry": "string",
  "receiverNationality": "string",
  "receiverIdType": "string",
  "receiverIdNumber": "string",
  "receiverEmail": "string",
  "receiverNativeFirstname": "string",
  "receiverNativeMiddlename": "string",
  "receiverNativeLastname": "string",
  "receiverAccountType": "string",
  "calcBy": "string",
  "transferAmount": "string",
  "remitCurrency": "string",
  "payoutCurrency": "string",
  "paymentMode": "string",
  "bankName": "string",
  "bankBranchName": "string",
  "bankBranchCode": "string",
  "bankAccountNumber": "string",
  "swiftCode": "string",
  "promotionCode": "string",
  "beneficiaryType": "string",
  "remitterType": "string",
  "senderNativeAddress": "string",
  "receiverNativeAddress": "string",
  "receiverOccupation": "string",
  "receiverOccupationRemarks": "string",
  "senderOccupationRemarks": "string",
  "senderIdTypeRemarks": "string",
  "receiverIdTypeRemarks": "string",
  "senderSourceOfFundRemarks": "string",
  "senderBeneficiaryRelationshipRemarks": "string",
  "purposeOfRemittanceRemarks": "string",
  "receiverIdIssueDate": "string",
  "receiverIdExpireDate": "string",
  "receiverDistrict": "string",
  "receiptCpf": "string",
  "remarks": "string"
}
```

Field	Data Type	Max Length	Description	Required
agentSessionId	String	150	Partner ID, WS System will response same agent session id to track the response is valid from same session	y
agentTxnId	String	20	Partner Unique TXN ID for each transaction.	Y
locationId	String	10	Payout Location ID/Bank ID Refer to *GetAgentList* method	y
RemitterType	String	1	I – Individual B – Business	y
senderFirstName	String	60	Sender First Name	y
senderMiddleName	String	60	Sender Middle Name	n
senderLastName	String	60	Sender Last Name **only if Compliance defines as mandatory*	*
senderGender	String	10	**Male* Female **only if Compliance defines as mandatory*	*
senderAddress	String	100	Sender Full Address	y
senderCity	String	50	Sender City **only if Compliance defines as mandatory*	*
senderState	String	50	Sender State, if passed then *Refer to* ***GetCatalogue****function CATALOGUETYPE:STA*	*
senderZipCode	String	15	Sender Zipcode **only if Compliance defines as mandatory*	*

senderCountry	String	3	ISO -3 Country Code For Example: PHL for Philippines	*
			*Refer **GetCatalogue** function* *CATALOGUETYPE: CTY* **only if* *Compliance defines as mandatory*	
senderMobile	String	16	Sender Mobile No, if valid then SMS notification will be delivered to Remitter when TXN is paid Provide Mobile Number with country code for SMS Notification	*
senderNationality	String	3	ISO -3 Country Code For Example: PHL for Philippines *Refer **GetCatalogue** function* *CATALOGUETYPE: CTY*	*
senderIdType	String	2	Sender Valid ID Type e.g. Passport No/Social Security *Refer **GetCatalogue** function* *CATALOGUETYPE: DOC*	*
senderIdTypeRemarks	String	60	Remarks for senderIdType if IdType inputted as Others	*
senderIdNumber	String	20	Sender ID Number	*
senderIdIssueCountry	String	3	ISO -3 Country Code For Example: PHL for Philippines *Refer **GetCatalogue** function* *CATALOGUETYPE: CTY* **only if Compliance defines as mandatory*	*
senderIdIssueDate	String	10	ID Issue Date format: yyyy-dd-mm **only if Compliance defines as mandatory*	*
senderIdExpireDate	String	10	Expire Date format yyyy-mm-dd **only if Compliance defines as mandatory*	*
senderDateOfBirth	String	10	Date of Birth Format yyyy-mm-dd **only if Compliance defines as mandatory*	*
senderOccupation	String	2	Sender Occupation *Refer to **GetCatalogue*** *function CATALOGUETYPE:* *OCC* *and pass DATA returned.* **only if Compliance defines as mandatory*	n
senderOccupationRemarks	String	60	Remarks for senderOccupation if senderOccupation inputted as Others	*

senderSourceOfFund	String	2	Sender Source of Fund *Refer **GetCatalogue** function* *CATALOGUETYPE: SOF* *and pass DATA returned.*	*

			**only if Compliance defines as mandatory*	
senderSourceOfFundRe mar ks	String	60	Remarks for senderSourceOfFund if senderSourceOfFund inputted as Others	*
senderSecondaryIdType	String	2	Sender Valid Secondary ID Type e.g. Passport No/Social Security *Refer to **GetCatalogue** function CATALOGUETYPE: DOC* **only if Compliance defines as mandatory*	*
senderSecondaryIdNumb er	String	20	Secondary ID Number **only if Compliance defines as mandatory*	*
senderEmail	String	50	Sender Email address **only if Compliance defines as mandatory*	*
senderNativeFirstName	String	60	Native first name of the sender **only if Compliance defines as mandatory*	*
senderNativeMiddleNam e	String	60	Native middle name of the sender **only if Compliance defines as mandatory*	*
senderNativeLastName	String	60	Native last name of the sender **only if Compliance defines as mandatory*	*
senderBeneficiaryRelatio ns hip	String	2	Relationship between sender and beneficiary *Refer to **GetCatalogue** function CATALOGUETYPE: REL* *and pass DATA returned.* **only if Compliance defines as mandatory*	Y
senderBeneficiaryRelatio ns hipRemarks	String	60	Remarks for senderBeneficiaryRelationship, if senderBeneficiaryRelationship inputted as Others	*
purposeOfRemittance	String	2	Purpose of Remittance *Refer **GetCatalogue** function* *CATALOGUETYPE: POR* *and pass DATA returned.* **only if Compliance defines as mandatory*	Y
purposeOfRemittanceR ema rks	String	60	Remarks for purposeOfRemittance if purposeOfRemittance inputted as Others	*

BeneficiaryType	String	1	I – Individual B – Business	y
receiverFirstName	String	60	Receiver First Name	Y
receiverMiddleName	String	60	Receiver Middle Name	n

receiverLastName	String	60	Receiver Last Name **only if Compliance defines as mandatory*	*
receiverAddress	String	100	Full Address **only if Compliance defines as mandatory*	*
receiverContactNumber	String	16	Receiver Contact Information **only if Compliance defines as mandatory*	*
receiverState	String	60	*Refer* ***GetCatalogue*** *function CATALOGUETYPE:* ***STA*** *and pass DATA returned.* **only if Compliance defines as mandatory*	*
receiverAreaTown	String	60	*Refer* ***GetCatalogue*** *function CATALOGUETYPE:* ***ART*** *and pass DATA returned.* **only if Compliance defines as mandator*	*
receiverCity	String	50	Receiver City **only if Compliance defines as mandatory*	*
receiverCountry	String	3	ISO -3 Country Code For Example: PHL for Philippines *Refer* ***GetCatalogue*** *function CATALOGUETYPE: CTY* **only if Compliance defines as mandatory*	y
receiverIdType	String	2	*Receiver ID Type (National ID, Driving License, etc)* *Refer* ***GetCatalogue*** *function CATALOGUETYPE: DOC* **only if Compliance defines as mandatory*	*
receiverIdTypeRemarks	String	60	Remarks for receiverIdType if receiverIdType inputted as Others	*
receiverOccupation	Sting	2	Receiver Occupation *Refer to* ***GetCatalogue*** *function CATALOGUETYPE: OCC and pass DATA returned.* **only if Compliance defines as mandatory*	n
receiverOccupationRemarks	String	60	Remarks for receiverOccupation if receiverOccupation inputted as Other	*
receiverIdNumber	String	50	Receiver ID Number	n
receiverEmail	String	50	Receiver Email address	
receiverNativeFirstName	String	60	Native first name of the receiver	*

receiverNativeMiddleName	String	60	Native middle name of the receiver	*
receiverNativeLastName	String	60	Native last name of the receiver	*
calcBy	String	1	P – Calculate by Payout Currency	y
transferAmount	Money		Pass the total Amount to be deposited/received by beneficiary/receiver in the payout country currency	y

remitCurrency	String	3	ISO3 Send Amount Currency Code	Y
payoutCurrency	String	3	ISO3 Payout Amount Currency Code	y
paymentMode	String	1	B- Account Deposit C- Cash Pickup	y
			P- Corporate Payment R- Card Payment W- Mobile Wallet V-Individual to Individual Bank Transfer X-Individual to Business Bank Transfer Y-Business to Individual Bank Transfer Z- Business to Business Bank Transfer *Refer to **GetCatalogue** function* *CATALOGUETYPE: PTY*	
bankName	String	150	Beneficiary Bank Name	n
bankBranchName	String	150	Beneficiary Bank branch name *In the case of India, pass IFSC Code	*
bankBranchCode	String	10		
bankAccountNumber	String	50	Beneficiary Bank Account Number, *Mandatory if PAYMENTMODE is B	*
swiftCode	String	10	Bank Swift Code if payout partner requires	n
promotionCode	String	50	Based on the Promotion code provided, System will return premium rate and fee discount	n
SenderNativeAddress	String	100	Multilingual characters for Sender Native Address (eg. काठमाड ,○ बागमती)	n
ReceiverNativeAddress	String	100	Multilingual characters for Receiver Native Address (eg. 加德满都,巴格马蒂)	n
receiverIdIssueDate	String	10	ID Issue Date format: yyyy-dd-mm **only if Compliance defines as mandatory*	*
receiverIdExpireDate	String	10	ID Issue Date format: yyyy-dd-mm **only if Compliance defines as mandatory*	*
receiverDistrict	String	100	Receiver District	n
receiptCpf	String	50	Receipt CPF *Mandatory for Brazil payout	*
remarks	String	100	Remarks	n

Response:

```
{
        “code”:”0”,
        "agentSessionID":”string”,
        “message”:”string”,
        “confirmationId”:”string”,
        “agentTxnId”:”string”,
        “collectAmount”:”Money”,
        “collectCurrency”:”string”,
        “serviceCharge”:”Money”,
        “gstCharge”:”Money”,
        “transferAmount”:”Money”,
        “exchangeRate”:”Float”,
        “payoutAmount”:”Money”,
        “payoutCurrency”:”string”,
        “feeDiscount”:”Money”,
        “additionalPremiumRate”:”Float,

         “txnDate”:”string”,

       “settlementRate”:”Float”,
       “sendCommission”:”Money”
       “settlementAmount”:”Money”

}
```

Field	Description
code	0 if Success otherwise refer to Error Code
agentSessionId	Partner Session ID
confirmationId	Unique Confirmation ID, which needs to be passed at the time of calling ***CommitTransaction*** to commit transaction.
agentTxnId	Partner Unique TXN ID for each transaction.
collectAmount	Total Amount to be collected from Customer including service fee
collectCurrency	ISO 3 Remit Currency Code e.g. MYR
serviceCharge	Service Fee in Collect Currency
gstCharge	Tax Charge on Service Fee if applicable
transferAmount	Total Remit value amount (excluding Service Fee)
exchangeRate	Exchange Rate
payoutAmount	Total Payout Amount to Payout Location
payoutCurrency	ISO 3 Payout Currency
feeDiscount	If PROMOTIONCODE has Fee Discount, it return Fee Discounted Value
additionalPremiumRate	If Promotion Code has Premium Rate, it will return Additional Premium Rate
txnDate	Transaction Date and Time as recorded as of time
settlementRate	The Rate with which Partner Settlement Amount is Calculated.
sendCommission	The Partner Commission for Sending Transaction in Collect Currency.
settlementAmount	Total Amount that Partner has to settle for this Transaction in Collect Currency

Note – After *SendTransaction*, you must call *CommitTransaction* to commit the transaction in the system. Session will expire in 30 mins. All those transactions which are not committed, will be truncated from system after one hour.

2.7 CommitTransaction

Call this method to Commit Transaction. After *SendTransaction*, you must call *CommitTransaction* to make transaction available. Session will expire in 30 mins. All those transactions which are not Authorized, will be truncated from system at the End of Day.

When Transaction is already Authorized and *CommitTransaction* is *re-invoked* using Same Confirmation
ID, it will return <CODE>100</CODE> which means Transaction has been successfully accepted and authorized.
To get real Status *QueryTXNStatus* has to be called.

Request URL
POST {connectbaseurl}/api/webservice/CommitTransaction

Request:
```
{
  "confirmationId":
  "string",
  "agentSessionId":
  "string",
  "remitCurrency": "string"
}
```

Field	Data Type	Max Length	Description	Required
confirmationId	String	16	Unique CONFIRMATIONID which you retrieve at the time of calling *SendTransaction*	Y
agentSessionId	string	150	WS System will response same agent session id to track the response is valid from same session	Y
remitCurrency	String	3	ISO3 Send Amount Currency Code	* ***Mandatory if partner account is setup as multiple sending currency account***

Response:
```
{
   "code":"0",
  "agentSessionId":"string",
  "message":"string",
  "pinNumber":"string",
  "senderName":"string",
  "collectAmount":"Money",
  "collectCurrency":"string",
  "payoutAmount":"Money",
  "payoutCurrency":"string",
  "status":"string",
  "isPinOnHold":"string"
    }
```

Field	Data Type	Max Length	Description
code	string	4	if "0" Success, else refer to Error Code
agentSessionId	String	Agent Session ID, which was send at the time of Request	
message	String	100	System Message
pinNumber	String	16	Random Unique Transaction PINNO which is used for beneficiary to pick up cash.
senderName	String	180	Full Sender Name
collectAmount	Money		Total Collected Amount in Sender Currency including fee
collectCurrency	String	3	ISO 3 Remit Currency
payoutAmount	Money		Total Amount to be Paid to Beneficiary in Payout CCY
payoutCurrency	String	3	ISO 3 Payout Currency
status	String	15	After Authorization Transaction is successfully Received and Status will be either - HOLD or - SANCTION or - COMPLIANCE Call *QueryTXNStatus* to get the real status.
isPinOnHold	String	5	- True - False If True – PINNO can NOT be provided to Customer. To retrieve FINAL PINNO, *QueryTXNStatus* has to be called (re-tried) until its Response returns <STATUS>Post</STATUS> and NEW PIN will be in <PINNO> If False – PINNO returned FINAL PIN and can be provided to Customer.

2.8 QueryTXNStatus and Callback

There are two ways to get the status of the transaction. One way is to call the QueryTXNStatus to get the status of the transaction. The other way is that the system will send the status of the transaction in a call back URL of the sending agent.

2.8.1 QueryTXNStatus

This method is used to the check the current status of transaction by PINNO or by Agent TXN ID.

Request URL

POST

{connectbaseurl}/api/webservice/QueryTXNStatus

Request:

```
{
 "pinNumber": "string",
 "agentSessionId": "string",
 "agentTxnId": "string",
 "remitCurrency": "string"
}
```

Field	Data Type	Max Length	Description	Required
pinNumber	String	16	You can make a query by PINNO which was response at the time of calling CommitTransaction	Y*
agentSessionId	String	150	WS System will response same agent session id to track the response is valid from same session	Y
agentTxnId	String	50	You can make a query by Partner unique Agent TXN ID which was send at the time of calling SendTransaction	Y*
remitCurrency	String	3	ISO3 Send Amount Currency Code	** Mandatory if partner account is setup as multiple sending currency account*

*You can make a query either by pinNumber or agentTxnId

*If you pass both, then error message will response

Response:

```
{
    "code": "0",
    "agentSessionId": "string",
    "message": "string",
    "pinNumber": "string",
    "senderName": "string",
    "receiverName": "string",
    "collectAmount": "Money",
    "collectCurrency": "string",
    "serviceCharge": "Money",
    "gstCharge": "Money",
    "transferAmount": "Money",
    "exchangeRate": "Float",
    "payoutAmount": "Money",
    "payoutCurrency": "string",
    "settlementAmount": "Money",
    "sendCommission": "Money",
    "settlementRate": "Float",
    "status": "string",
    "statusDate": "string",
    "additionalMessage":
    "string", "agentTxnId":
    "string",
}
```

Field	Data Type	Max Length	Description
code	string	4	if "0" Success, else Error
agentSessionId	String	150	Agent Session ID, which was send at the time of Request
message	String	100	System Message
pinNumber	String	16	PIN NO, which is used to payout transaction for Beneficiary
senderName	String	180	Full Sender Name
receiverName	String	180	Full Receiver Name
collectAmount	Money		Send Amount including fee
collectCurrency	String	3	ISO 3 Send Currency
exchangeRate	Float		Exchange rate applied for this transaction
serviceCharge	Money		Service Charge applied for this transaction
gstCharge	Money		Tax on Service Fee
transferAmount	Money		Transfer amount excluding fee
payoutAmount	Money		Total Amount to be Paid to Beneficiary in Payout CCY
payoutCurrency	String	3	Payout Currency Code

settlementAmount	Money		Settlement Amount
sendCommission	Money		Partner Commission for Sending Transaction, shared Service Charge
settlementRate	Float		The Rate at which Settlement Amount is calculated with Partner
status	String	30	Status of Transaction - UN-COMMIT-HOLD - UN-COMMIT-COMPLIANCE - HOLD - COMPLIANCE - SANCTION - UN-PAID - POST - PAID - CANCEL - CANCELHOLD - API PROCESSING - BLOCK
additionalMessage	String	300	Status Message from Payout Partner
statusDate	DateTi me	20	Paid Date or Cancel Date else it will be Blank
agentTxnId	String	20	Unique TXN ID which was passed at the time of calling SendTransaction

2.8.2 Callback function

After the sending partner's call back URL is implemented in the RemitX system -

What the sending partner should expect?

At sending partner's callback URL end point the system will send HTTP Post request with JSON body and authorization header: (Similar to section 1.3)
[Authorization: hmacauth APPId:Signature:Nonce:Timestamp]

We recommend for sending partner to validate the request to determine if it is being sent from the RemitX system.

Request:

```
{
  " agentTxnId ":
  "string", " pinNumber":
  "string", " status":
  "string",
  " message": "string",
  " notification_ts": "string"
}
```

Field	Description

agentTxnId	agentTxnId as sent by merchant in SendTransaction request
pinNumber	pinNumber as received at partner’s CommitTransaction response
status	Status of transaction (either **Paid** or **Cancel**)
message	Message for the transaction
notification_ts	Notification for status update

What the RemitX system will expect?

- HTTP 200 OK if everything is good to go sending partner’s end
- Any other Status code for exceptional case at the sending partner’s end

<u>Note for non 200 OK Status code</u>

- If other status code is received at our end then we re-send the request two more times (total 3 times including the initial request) until 200 OK is received.
- If after 3rd attempt, RemitX system still receives status other than 200 then an email is sent to the sending partner’s email address which is setup in the system.
- After this, the sending agent needs to call QueryTXNStatus manually to inquire status of the transaction.

2.9 CheckBalance

Call this method to find partner current balance.

webservice/

POST {connectbaseurl}/api/webservice/CheckBalance

Request:

```
{
        "agentSessionId": "string",
        "remitCurrency": "string"
}
```

Field	Data Type	Max Length	Description	Required
agentSessionId	String	150	WS System will response same agent session id to track the response is valid from same session	Y
remitCurrency	String	3	ISO3 Send Amount Currency Code	* ***-Mandatory in case of Multiple Sending CCY only***

Response:

```
{
    "code": "0",
    "agentSessionId": "string",
    "currentBalance": "string",
    "availableLimitBalance":
    "string", "currency": "string"
}
```

Field	Data Type	Max Length	Description
code	string	4	if “0” Success, else Error
agentSessionId	String	150	Agent Session ID, which was send at the time of Request
message	String	100	Transaction Notes if any
currentBalance	Money		Partner Current Balance
availableLimitBalance	Money		Available Limit to Send Transaction, Available Limit depends on Credit Limit assigned to Partner and the Current Balance.
currency	String	3	Current Balance and Available Limit Balance ISO 3 Currency Code

2.10 GetFieldInfo

Partner call *GetFieldInfo* Function to get mandatory and optional field for Send Transaction.

Request URL

POST {connectbaseurl}/api/webservice/GetFieldInfo

Request:

```
{
  "agentSessionId":
  "string", "partnerId":
  "string", "locationId":
  "string", "payoutCountry":
  "string",
  "payoutCurrency":
  "string", "paymentMode":
  "string",
  "transferAmount":
  "string",
  "remitCurrency": "string"
}
```

Field	Data Type	Length	Description	Required
agentSessionId	string	150	Partner ID, WS System will response same agent session id to track the response is valid from same session	y
locationId	String	10	Payout Branch Location ID (Branch ID) or BANK ID Refer to *GetAgentList* method	y
payoutCountry	String	3	ISO -3 Country Code For Example: PHL for Philippines *Refer **GetCatalogue** function CATALOGUETYPE: CTY*	y
payoutCurrenc y	String	3	ISO3 Payout Amount Currency Code	y
paymentMode	String	1	C- Cash Pickup B- Account Deposit P- Corporate Payment R- Card Payment W- Mobile Wallet V-Individual to Individual Bank Transfer X-Individual to Business Bank Transfer Y-Business to Individual Bank Transfer Z- Business to Business Bank Transfer	y
transferAmoun t	Money		Amount to be deposited/received by beneficiary/receiver in the payout country currency	y
remitCurrency	String	3	ISO3 Send Amount Currency Code	** **Mandatory if partner account is setup as multiple sending currency account***

Response:

```
{
    "CODE": "string",
    "agentSessionId": "string",
    "message": "string",
    "fieldList" : [
      {
            "fieldName":"string",
            "fieldLabel":"string",
            "required":"boolean",
            "dynamicField":"boolean",
            "minLength":"integer",
            "maxLength":"integer"
      },
            {
            "fieldName":"string",
            "fieldLabel":"string",
            "required":"boolean",
            "dynamicField":"boolean",
            "minLength":"integer",
            "maxLength":"integer"
      }
    ]
}
```

Field	Data Type	Description
code	String	0 if Success otherwise Error Code
agentSessionId	String	Partner Session ID
message	String	Message to Partner, for example Payout Location detail
fieldList	Object List	
fieldName	String	API field name
fieldLabel	String	API field label
required	Boolean	API field required (true/false)
dynamicField	Boolean	Dynamic field in API (true/false) *If true then **FIELDNAME** and **FIELDVALUE** must be passed in **SendTransaction.DYNAMICFIELD** object.* *If false no need add in **SendTransaction.DYNAMICFIELD** object.*
minLength	Integer	API field value minimum length
maxLength	Integer	API field value maximum length

Note: Fields – **RemitterType & **BeneficiaryType** – are not listed in GetFieldInfo Response, however, these two fields are default mandatory*

LIGHTNET

Appendix 1 – Codes and Description

CODE ID	MESSAGE
0	Success
1	Transaction is already authorized. Check Query Transaction Status to find the detail status
2	Transaction is created but requires CommitTransaction
100	Transaction is already accepted and authorized
1001	Mandatory fields are blank
1002	Authentication Fail
1003	User Is Locked
1004	Invalid Parameter
1005	Duplicate Agent TXN ID
1006	Invalid Field Name
1007	Invalid Field Value
1008	Invalid Field Value Length
1010	Invalid Data Format
1099	No Data Found
1100	Signature is invalid
1200	Field maximum length exceeded.
2001	Transaction is not in Authorized Mode
2003	Invalid Transaction
3001	Invalid Payment Mode
3002	Invalid Bank ID
3003	Invalid Location ID
3004	Location ID is inactive
3005	Agent TXNID does not match
3006	Agent TXN ID is missing
3007	Transaction Cancelled Error
3008	Select Country is not allowed
3009	Sent Amount must be more than Service Charge
3010	Bank Transfer Account No is Blank
3011	Payout Amount Limit Exceed
3012	Partner Balance Exceed, Cannot Make a TXN
3013	No Record Found
3014	Selected Location is inactive
4011	Catalogue is blank
4014	Invalid request parameter
4016	No data found
4019	#BankName# #AgentInactive#
4020	Invalid Location ID and Country

4021	Select country is not allowed, please contact Head Office
4022	Service Charge is NOT defined for requested Remittance Amount!

4023	Collected amount is invalid must be more than #ServiceFee#
4024	Invalid Payout Currency
4024	Maximum Transaction amount has exceeded
4025	Amount is invalid
4029	TXN limit exceeded. You have limit up to #LimitPertxn# #CollectCurrency# #PerTransaction#
4032	Receiver country is mandatory
4040	Selected Payment Mode is not available for country #PayoutCountry#
4051	Sender Name missing
4052	SENDER_ADDRESS is mandatory
4060	Sender Identity Type missing
4063	Receiver Name missing
4064	Receiver Country missing
4067	SENDER_ID_ISSUE_DATE is mandatory for selected PaymentType
4069	SENDER_IDENTITY_TYPE did not match.
4071	Payment Type missing
4072	Transfer Amount must be numeric value
4073	LOCATION ID is missing
4074	Transfer currency is required
4075	Location Id is not provided or Location ID doesnt matched with PayoutCountry
4079	SENDER_ID_EXPIRE_DATE is mandatory or selected PaymentType
4082	CALC_BY missing or Invalid CALC_BY
4083	Agent SESSION ID is missing
4084	Sender ID Issue Date is invalid Must be YYYY-MM-DD
4085	Sender ID Expire Date is invalid Must be YYYY-MM-DD
4086	Sender Date of Birth is invalid Must be YYYY-MM-DD
4087	Duplicate Agent TXN ID: #AgentTranID#
4088	Location ID is not active
4089	For Bank Transfer Location ID and Bank ID must be defined
4090	Payout BANK ID is invalid
4091	Bank Account No is blank
4095	Payout Branch ID is invalid
4095	Delivery Option not defined
4095	Delivery Option not defined
5004	Agent TXN ID is missing
5006	Invalid Transaction
5008	Technical Error
5011	Receiver ID is required

5012	Correct receiverID is required.

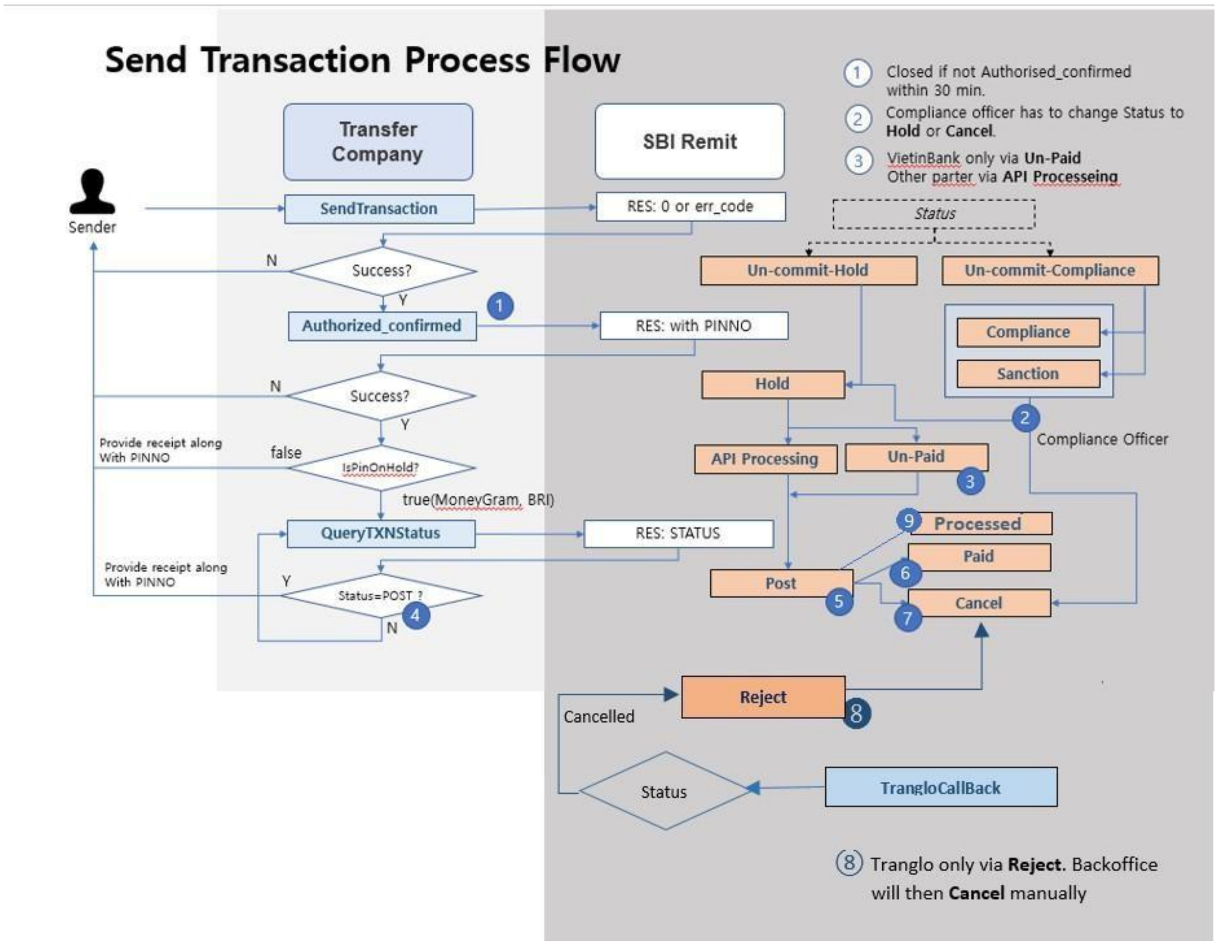
5043	PINNo: #PINNO# cannot amend. Please contact Headoffice
5044	Cannot cancel. Please contact Headoffice
5085	Invalid From Date must be YYYY-MM-DD and for Time HH:MM:SS (H24)
5086	Invalid Date Type, Must be a - List all TXN s - Trans Send Only P - Trans Paid Date C - List all Cancel Trans u - List Un-Paid TXN only
5087	Invalid To Date must be YYYY-MM-DD and for Time HH:MM:SS (H24)
5088	Invalid ISO3 Country code in Payout Country
5089	Invalid Payment Type, must be C-Cash Pickup, B-Account Deposit, P-Corporate Payment, R-Card Payment, W-Mobile Wallet
5092	Invalid Confirmation ID
5095	Invalid ISO3 Country code in Sender Country
5096	Invalid Sender Nationality ISO3 Code
5097	Invalid ISO3 Country code in Sender ID Issue Country
5099	You do not have sufficient balance to send txn
5100	Invalid ISO3 Country code in Additional_field1
5101	For Catalogue#CATALOGUE_TYPE# (Payment Mode), in ADDITIONAL_FIELD1 ISO3 Country Code must be provided
5102	Max Send Amount Limit#sendlimit# #COLLECTCURRENCY#
5103	Max Payout Limit Amount is#maxpayoutamt# #PayOutCur# for Country: #Country#
5104	Remit Currency is missing
5105	Payout Currency is missing
5106	PURPOSE_OF_REMITTANCE is invalid.
5108	Service Charge is Not Defined for select Payout Amount
5109	Service Charge is Not Defined for the Remit Amount
5110	you are not allowed to remit currency#remitcurrency#
5113	Amount you entered is not valid.
5120	Sender Gender is Invalid
5121	Sender Occupation does not Match
5122	Calculate Method is Invalid
5124	Bank or Location ID does not support account validation
5126	Receiver ID Type does not match
5127	Sender Source of Income is Invalid
6000	Payout locations for the requested country is not available. Please contact Lightnet support.
6001	Invalid country
6002	Invalid currency

6004	Invalid date related messages - Sender Id issue date invalid (Already expired) - Sender Id issue date invalid

	- Date of birth invalid (must be 18 years)
7000	Service not available at the moment
9001	Technical Error

Appendix 2 – Send Transaction Flow

The Send Transaction Process showing Transaction STATUS in each event.



Appendix 3 - Transaction STATUSES

UN-COMMIT-HOLD

Transaction has been successfully saved using *SendTransaction* and requires *CommitTransaction* within 30 minutes.

UN-COMMIT-COMPLIANCE

Transaction has been successfully saved using *SendTransaction* and requires *CommitTransaction* within 30 minutes. Upon CommitTransaction, the Transaction Status will either be Compliance or Sanction.

HOLD

Upon calling *CommitTransaction* the Un-Commit-Hold Transaction Status changes to Hold Status.
This Transaction is now ready for API Auto Scheduler which will push Transaction to Partner System.

COMPLIANCE

Transactions which fall under Compliance screening against High Volume and Frequency, Duplication (same Sender, Receiver and Amount on same Day) are in Compliance Status after CommitTransaction. Compliance Officer has to release Transactions after which the Transaction Status is updated as Hold.

SANCTION

Transactions where Sender and Receiver Names match with Sanction List Entries (OFAC, PEP, and Police etc.) are in Sanction Status after *CommitTransaction*. Compliance Officer has to release Transactions after which the Transaction Status is updated as Hold.

UN-PAID

The Transactions are in Un-Paid Status after Manual or Auto Scheduler Authorization. The Un-Paid Status is limited to Non-API Payout Partner Transactions only. That means transactions whose destination Payee calls Pay WebServices or uses Agency Panel to pay Transaction.

API PROCESSING

The API Processing Status is returned in cases when the Transaction Fails at API Payout Partner due to technical reasons (For example time out). Payment Ops team will handle such cases.

POST

Once the Transaction is pushed successfully to API Payout Partner System, the Transaction is in Post Status. This means the Transaction is available for payment or to ready to be credited at partner system. Amendment and Cancellation can be performed by Head Office or Admin only.

PAID

Once the Transaction has been successfully disbursed to Final Beneficiary (cash payout, credit to Bank) the Transaction is in PAID status.

CANCEL

The Transaction is in CANCEL status if it was cancelled via API.

CANCELHOLD

The Transaction is in CANCELHOLD status if it was cancelled by Head office/Admin in HOLD status.

BLOCK
Transactions that are suspended/blocked manually by Admin/Head Office are in Block Status. Blocking will not cancel the transaction, but prevent it from being released for payment. Admin/ Head Office may decide to Block/ Unblock Transaction based on Funding Issues with Send Partner or Customer/beneficiary. The Transaction can be manually updated to Unblock (previous Status).

Appendix 4 – Sample code for Signature generation

```
public string GenerateHMACSignature () {
    string signaturestring = string.Empty;
    string appId = "provided_app_id";
    string appKey = "provided_app_key";
    string apiUrl = "https://staging-api-lqn.lightnet.io/api/webservice/getecho";
    string method = "post";
    string requestBody = @"{
" + "\n" +
        @"  ""agentSessionId"": ""1622024176""
" + "\n" +
        @"}";
    // HttpResponseMessage response = null;
    string requestContentBase64String = string.Empty;
    //Get the Request URI
    string requestUri = HttpUtility.UrlEncode (apiUrl.ToLower ());
    //Get the Request HTTP Method type
    string requestHttpMethod = method.ToUpper (); //request.Method.Method;
    //Calculate UNIX time
    DateTime epochStart = new DateTime (1970, 01, 01, 0, 0, 0, 0, DateTimeKind.Utc);
    TimeSpan timeSpan = DateTime.UtcNow - epochStart;
    string requestTimeStamp = Convert.ToUInt64 (timeSpan.TotalSeconds).ToString ();
    //Create the random nonce for each request
    string nonce = Guid.NewGuid ().ToString ();
    //Checking if the request contains body, usually will be null wiht HTTP GET and DELETE
    if (requestBody != null) {
        // Hashing the request body, so any change in request body will result a different hash
        // we will achieve message integrity
        //byte[] content = await request.Content.ReadAsByteArrayAsync();
        byte[] content = Encoding.UTF8.GetBytes (requestBody);
        MD5 md5 = MD5.Create ();
        byte[] requestContentHash = md5.ComputeHash (content);
        requestContentBase64String = Convert.ToBase64String (requestContentHash);
    }
    //Creating the raw signature string by combinging
    //APPId, request Http Method, request Uri, request TimeStamp, nonce, request Content Base64 String
    string signatureRawData = String.Format ("{0}{1}{2}{3}{4}{5}", appId, requestHttpMethod, requestUri,
        requestTimeStamp, nonce, requestContentBase64String);
    //Converting the APIKey into byte array
    var secretKeyByteArray = Convert.FromBase64String (appKey);
    //Converting the signatureRawData into byte array
    byte[] signature = Encoding.UTF8.GetBytes (signatureRawData);
    //Generate the hmac signature and set it in the Authorization header
    using (HMACSHA256 hmac = new HMACSHA256 (secretKeyByteArray)) {
        byte[] signatureBytes = hmac.ComputeHash (signature);
```

```
string requestSignatureBase64String = Convert.ToBase64String (signatureBytes);
```

```
        //Setting the values in the Authorization header using custom scheme (hmacauth)
        signaturestring = string.Format ("{0}:{1}:{2}:{3}", appId, requestSignatureBase64String, nonce,
requestTimeStamp);
        // request.Headers.Authorization = new AuthenticationHeaderValue("hmacauth", signaturestring));
    }
    return signaturestring;
 }
```