Protocol for Sponsor Bank API

Initial Public Offer System

Version 0.15.0

Oct 2024

National Stock Exchange of India Ltd
Exchange Plaza, Plot No. C/1, G Block,
Bandra-Kurla Complex, Bandra (E),
Mumbai - 400 051.

Revision History

Date	Change Description	Edited By	Version
08-Oct-2018	Draft Version for review		0.9.1
09-Oct-2018	Review changes		0.9.2
09-Nov-2018	Added Sponsor Bank File Interface. End of Issue Reconciliation file		0.9.3
06-Dec-2018	Added Sponsor Bank API to send addition / modifications in the IPO Master. Added “sbReason“ field in “RTA Fund Transfer / Release Response File”		0.9.4
24-Dec-2018	No Change		0.9.5
24-Dec-2018	Corrected widths of fields “sym”, “appNo” and fields related to “seqNo” across API’s and file structures Extensions of files in RTA interface changed to “.zip” Added “payStatus” value of 13 in Sponsor bank API POST /v1/clientapplicationstatus		0.9.6
24-Jul-2019	Change in API POST /v1/clientapplication : • Request Json : Change in description of field "cat". Page No: 7. • Bid Detail Json : Change in descriptions of fields "qty" & "amt". Page No: 8. Change in width of field "sbRefNo" & "payRefNo" in • Response Json of API POST /v1/clientapplication. Page No: 8. • Client Application Status Json (Response) of API POST /v1/clientapplicationstatus. Page No: 10. • "Data Record Structure" of "End of Issue Reconciliation file". Page No: 16. • "Data Record Structure" of "RTA Fund Transfer/Release file". Page No: 18. • "Data Record Structure" of "RTA Fund Transfer/Release Response file". Page No: 19. Change in API POST /v1/ipomaster • Request Json : Change in descriptions of fields "name" & "paymentAcceptanceTime". Page No: 12. Change in "End of Issue Reconciliation file" • New fields "dpId","benId" & "pan" added in "Data Record Structure". Page No: 15 & 16.		0.9.7
27-Aug-2019	• Introduction of New **payStatus** (i.e. 22) • Change in description of the field **rejectReason** • Introduction of additional field **umn** (Unique Mandate no rcvd in sponsor bank response) • Introduction of new field **umn** (Unique Mandate no) in **RTA Fund Transfer / Release file** • Added **Exchange-Sponsor Bank Reco File Interface**. End of day Issue Reconciliation file		0.9.8

17-Dec-2019	• IPO Master – Name. A prefix of "DB" or "EQ" would be appended to actual issue name to identify the type of issue (i.e. Debt or Equity) changed from DEBIT_ and EQUITY_. Page No 13 • Exchange-Sponsor Bank Reco File Interface. New columns added bidid, deleteflag. Page No 21 • Exchange-Sponsor Bank Reco File Interface. Note included. Page No 22 • Sponsor Bank API- clientapplication. Description changed for CliNm.Page No-8		0.9.9
16 Apr 2020	• IPO Master – Name. Removed prefix 'DB', 'EQ' • IPO Master – Added new column issueType • Sponsor Bank - Exchange Debit Revoke Status File added		0.10.0
15 Aug 2020	• API Communication Version v2: Change in Encryption/Decryption approach		0.11.0
28 Jan 2021	• API Communication Version v2: Changed ciper for RSA & Corrected AES key size to 256 bits in one place.		0.11.1
23 Aug 2021	• New Sponsor Bank API POST /<version>/clientapplicationbulk • New Exchange API POST /<version>/clientapplicationstatusbulk		0.12.0
10 Jan 2022	File nomenclature change in Exchange-Sponsor Bank Reco File Interface		0.12.1
12 May 2022	• New code 'REIN' for field issueType added in IPO master structure in API's POST /<version>/ipomaster and GET /<version>/ipomaster		0.13.0
20-Sep-2023	Change in length of field " accNo " - • Changes in "Client Application Status JSON" (Page 17) - Increase length of field accNo from 30 to 45 • Page 27 - Changes in Sponsor Bank File Interface (SB to RTA) - End of Issue Reconciliation file (RECO) - Increase length of field accNo from 30 to 45 • Page 29 - Changes in RTA Interface (RTA to SB) - RTA Fund Transfer / Release file - Increase length of field accNo from 30 to 45 • Page No 31- Changes in Exchange-Sponsor Bank Reco File Interface (Exchange to SB) Daily Issue Reconciliation file (RECO) - Increase length of field accNo from 30 to 45		0.14.0
11-Oct-2024	• Page 27 – Added in Sponsor Bank File Interface - Sponsor Bank -RNU File • Page 29 – Added in Sponsor Bank File Interface -Sponsor Bank – Debit Unblock File • Page 33 - RTA Interface – Allotment File		0.15.0

	• Page 35 – Added in Exchange-Sponsor Bank Reco File Interface - Closed User Group (CUG) • Page 36- Added SFTP File Path and Description • Removed section FNDSTAT_<exc>_<sym>_<sb>_<ddmmyyyy>.zip from Exchange-Sponsor Bank Reco File Interface - Exchange Debit Revoke Status File		

Preface

Purpose

This document describes the protocol for Web API based communication and messaging between exchanges and Sponsor Banks. The document also gives details on file based interface with the RTA's, Exchange and Sponsor Banks. The document serves as a common guide for developing and testing interfaces between systems at Exchanges, RTA and Sponsor Banks.

Target Audience

- Exchanges offering IPO platform
- Sponsor banks who intend to provide the payment service using UPI-2 for Retail IPO orders
- RTA – Registrar and Transfer Agents

Organization of This Document

This document is organized as follows:

Chapters	**Description**
Introduction	Overview of the document covering security features of the API and online messaging guidelines
Sponsor Bank API	This chapter gives details of the APIs to be exposed by the sponsor banks. These API's will be consumed by exchanges.
Exchange API	This chapter gives details of the API's exposed by exchanges. These API's will be consumed by sponsor banks.
Sponsor Bank File Interface	This chapter gives details of files exchanged between sponsor banks and exchange using SFTP service provided by exchange.
RTA Interface	This chapter gives details of the files being exchanged between RTA and sponsor banks using SFTP service provided by exchange
Exchange –Sponsor Bank Reco File Interface	This chapter gives details of files exchanged between exchange & sponsor banks using SFTP service provided by exchange for reconciliation on daily basis at Eod.

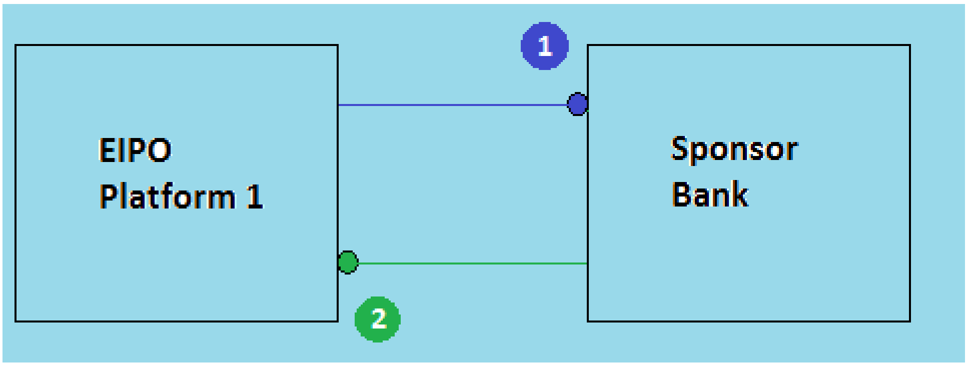
Table of Contents

Introduction

Exchange & Sponsor Banks will communicate through API as mentioned.

1. Sponsor banks will provide an API which will be consumed by bidding Exchange.
2. Bidding Exchange will provide an API which will be consumed by sponsored banks.

Both set of API's will be provided as REST endpoints accessible over internet using HTTPS protocol. All messages in the online API's will be in JSON format.



API Security

Exchange will generate a 2048-bit RSA key pair for every sponsor bank and will share the public key of the pair with the sponsor bank and securely keep the private key.
Similarly Sponsor bank will also generate a 2048-bit RSA key pair for every exchange and will share the public key of the pair with the exchange and securely keep the private key.

The APIs will support 2 modes of security

- Basic Security - Identified by API version "v1". Provides authentication of API invoker using asymmetric keys.
- Advanced Security - Identified by API version "v2". Provides authentication, encryption and data integrity checks.

The API version ("v1" or "v2") will be part of URL for all APIs (Exchange as well as Sponsor Bank). Exchange would host both versions of the API's which can be consumed by the sponsor bank. Sponsor bank will have to inform the exchange about the version of API being hosted by them.

Basic Security (v1)

- The API consumer application is expected to send following headers in every API HTTP request.

    - **LoginId**: Unique identifier to identify the entity invoking the API. The identifier for National Stock Exchange will be **NSE** and that for Bombay Stock Exchange will be **BSE.** In case of sponsor bank the 4 character bank code (first 4 characters used in banks IFSC) code will be used as identifier.
    - **Checksum**: The checksum will have to be generated by performing following steps.
        - Message payload will be first encoded using SHA256. In case of POST requests, the JSON message in the request body will be considered as payload. In case of GET methods, the “LoginId” will be considered as payload.
        - The encoded payload will then be encrypted using the public key shared by counter party using cipher transformation “RSA/None/PKCS1Padding”.
        - The binary encrypted contents will then be base64 encoded to convert it into ASCII text. This text will form the checksum for the request.
- The API provider will authenticate by performing following steps
    - The checksum will have to be base64 decoded first
    - The base64 decoded binary data will then have to be decrypted using self private key available with the API provider for the given “LoginId” using cipher transformation “RSA/None/PKCS1Padding”.
    - The message payload (message body in case of POST or value for header “LoginId” in case of GET) will be SHA256 encoded and then compared with the decryption output of the previous step. A match indicates successful authentication of the API invoker.
- The API provider **is not required** to provide any header (LoginId or Checksum or any other header for authentication) in its response.

Advance Security (v2)

Encryption

- The entity invoking the API should encrypt the request data (in case of POST method only) and add as request body in the HTTP request.
- The API response data should also be encrypted and sent as HTTP response.
- Encryption should be done in 2 steps.
- In first step the data should be encrypted using AES algorithm using 256-bit key.
- In second step the encrypted binary output should then be base64 encoded.
- The AES key should be randomly generated for every request or for a time window of say 15 minutes.
- The AES key should be encrypted using counter entity’s public key and then base64 encoded. The encrypted and encoded key should be inserted as a header in request or response. Header key name should be “**ipo-key-1**”.

Data Integrity

- The API invoker should insert “**ipo-key-2**” header in the HTTP request representing a message digest.

- In case of POST method the digest should be constructed by computing SHA256 of the request body followed by base64 encoding. Note that the request body contains encrypted and encoded data.

- In case of GET method which do not have a request body, a randomly generated string not more than 50 characters should be set as digest. Note that the digest would not be required for integrity check but will be required for authenticating the API invoker. (See below)

- In case of responses by API provider, the digest should be constructed in similar fashion using the encrypted and encoded response body and should be added in response header.

Authentication

- The entity (Exchange or Sponsor Bank) which is invoking an API will identify itself by providing a digital signature. A digital signature would comprise of a digest and signature obtained by encrypting the digest. Following steps should be performed.

- The API invoking entity should insert a header "**id**". This will be a unique identifier to identify the entity invoking the API. The identifier for National Stock Exchange will be NSE & for Bombay Stock Exchange will be BSE. In case of sponsor banks the 4 character bank code (first 4 characters used in banks IFSC) code will be used as identifier.

- Additionally a header "**ipo-key-3**" should also be inserted in the request. Signature should be computed by first encrypting the digest (computed for Integrity check) using entity's private key and then encoding the encrypted result using base64.

- The API responder is not expected to add headers "id" and "ipo-key-3" in the responses.

Decryption

- The HTTP request received by API provider should be decrypted.
- Additionally the HTTP response received by API invoker should also be decrypted.
- Header "**ipo-key-1**" should be read and decrypted first by base64 decoding the header value and then decrypting using the self private key corresponding to the counter entity. This will yield an AES key.
- The encrypted data should then be decrypted using the key and AES algorithm.

Verifying Authentication

- The API provider entity should authenticate API invoker entity using following steps
- Read the header "**id**" and fetch corresponding public key provided by the counter entity.
- Read the header "**ipo-key-3**" for signature. Base64 decode it and then decrypt it using the public key fetched in above step.
- Read the header "**ipo-key-2**" for digest and match it with the text decrypted in the above step. A successful match would indicate successful authentication.

Verifying Integrity

- Integrity verification should be performed on request body data received in POST method.
- Integrity verification should also be done for all API responses received from the API provider.
- A digest should be computed first by SHA256 encoding the original encrypted and encoded data received in request body or response and then base64 encoding the same.

- ➢ The computed digest should be matched with the header value “ipo-key-2” received. A successful match would indicate a successful integrity verification.

Summary

Header Key	Description	Computation Logic	Required in Request	Required in Response
id	Identifier for the entity invoking the API	NSE / BSE / Bank code	Yes	No
ipo-key-1	RSA Encrypted randomly generated AES key used for encrypting request payload or response	BASE64Encode(RSAEncrypt(AESKey, CounterPublicKey)	Yes	Yes
ipo-key-2	Message digest by hashing POST request payload or response. For GET request a random alphanumeric string (<=50 chars) should be set as message digest	BASE64Encode(SHA256(Encrypted POST Request/Response Payload))	Yes	Yes
ipo-key-3	Signature by RSA encrypting the message digest (ipo-key-2) using self private key. In case of GET request where no	BASE64Encode(RSAEncrypt(ipo-key-2, SelfPrivateKey))	Yes	No

Following key sizes and cipher transformations to be used

Algorithm	Key Strength	Cipher Transformation
RSA	2048	RSA/NONE/OAEPWithSHA1AndMGF1Padding
AES	256	AES/GCM/NoPadding

Message Format

1. Messages will be in JSON.
2. Following headers need to be provided in all API calls
   - **Content-Type**: Header value should be “application/json”
3. Path parameters and query parameters in the URL’s must be encoded using percentage encoding. (Refer http://www.w3schools.com/tags/ref_urlencode.asp for details)
4. All request and response messages are in JSON (Javascript Object Notation) format. (Refer http://www.json.org/ for details).
5. Some of the key specifications related to JSON and standards followed for the API’s are as follows
   - JSON is built on 2 structures. Object or Map containing key value pairs and an ordered list of values.
   - A value could be boolean (true / false), number, decimal, String or a structure (List or Object).
   - Object or key value pair structure consists of keys which are strings and values of any of the above types. E.g. {“name”:”Amit”, “age”:25}
   - List contains list of values. E.g. [“Amit”, “Ajay”, “Vikas”]

- A Boolean has only 2 values true or false.
- String values are enclosed in single quote or double quotes. e.g. “name”, “Amit”, “Pending”
- Numbers and decimals are represented without any thousand - separator character. Decimal indicator is dot (“.”)
- All dates, times and datetimes are represented as strings and in Indian standard time. Dates are formatted using format “dd-MM-yyyy”. Time are formatted as “hh24:mm:ss”. Date times are formatted as “dd-MM-yyyy hh24:mm:ss”.

6. All API’s can throw errors in the form of a common error response JSON

Field	Type	Mand.	Description
status	Number	Yes	1 = Message structure validation failures 2 = Business rule validation failures 3 = Any other error or unexpected 99 = Authentication failure.
msg	String(100)	Yes	Error message

```
{
"status":2,
"msg":"Invalid Symbol"
}
```

Sponsor Bank API

POST /<version>/clientapplication

This API will allow exchanges to submit IPO application request (new/modification/cancellation) to the sponsor bank.

Method	POST
Request	JSON
Response	JSON

Request JSON

Field	Type	Mand.	Description
exc	String(3)	Yes	Unique Identifier of Exchange (Same as LoginId in header)
sym	String(10)	Yes	Symbol
appNo	String(16)	Yes	Application No.(of the physical form) Symbol and Application No together uniquely identify a client application
seqNo	Number(9)	Yes	A unique sequence number for every message for a given exchange, symbol combination. The sequence number will always start with 1 for a given exchange and symbol and will be incremented exactly by 1. The sequence number will be useful in detecting gaps and filling them.
memCd	String(10)	Yes	Trading member code assigned by exchange
cat	String(5)	Yes	Alphanumeric Category Code of the Investor
cliNm	String(50)	No	Optional Client Name – Name of client. Exchange will remove the special characters while passing to sponsor bank, if any
dep	String(4)	Yes	Depository Name (NSDL, CDSL)
dpId	String(8)	No	DP ID (exactly 8 alphanumeric characters) Ignored in case of CDSL.
benId	String(16)	Yes	Client ID in case of NSDL (exactly 8 alphanumeric characters) Beneficiary ID in case of CDSL (exactly 16 alphanumeric characters)
pan	String(10)	Yes	Permanent Account Number
upi	String(45)	Yes	Universal payment identifier of the client
entTm	DateTime	Yes	Application entry time in the Exchange Bidding System
modTm	DateTime	No	Last modification time in the Exchange Bidding System

action	String(1)	Yes	N=New Application
			M=Modified Application
			R=Resend Application (for resending notification to Client)
			C=Cancelled Application

bids	List of Objects	Yes	List of bid details. See structure below. May be empty in case of Cancelled Application (actionCode = C)

Bid Detail JSON

Field	Type	Mand.	Description
bidId	Number(16)	Yes	Unique Identifier for the bid.
qty	Number	Yes	Bid Quantity. In case of DEBT ipo the sum of bid quantity of all series
pr	Decimal(10, 2)	Yes	Bid Price
amt	Decimal(15, 2)	Yes	Bid Amount. In case of DEBT ipo the sum of bid amount of all series

Sample Request

```
{
	"exc": "NSE",
	"sym": "TEST",
	"appNo": "1200299929020",
	"seqNo": 1,
	"memCd": "99999",
	"cat": "IND",
	"cliNm": "Sumit Aggarwal",
	"dep": "NSDL",
	"dpId": "IN012345",
	"benId": "12345678",
	"pan": "AFAKA2323L",
	"upi": "sumitagg01@upi",
	"entTm" : "01-10-2018 13:40:55",
	"action": "N",
	"bids": [
		{
			"bidId": 2018100100000122,
			"qty": 100,
			"pr": 55.30,
			"amt": 5530.00
		},
		{
			"bidId": 2018100100000123,
			"qty": 106,
			"pr": 55.00,
			"amt": 5830.00
		}
	]
}
```

Response JSON

Field	Type	Mand.	Description
exc	String(3)	Yes	Unique Identifier of Exchange (Same as LoginId in header)
sym	String(10)	Yes	Symbol
appNo	String(16)	Yes	Application No.(of the physical form) Symbol and Application No together uniquely identify a client application
status	Number	Yes	0 = Success Number greater than 0 = Failed. See section “Message Format” above for error status codes.
msg	String(100)	No	Error message in case of failure.
sbRefNo	String(50)	No	Sponsor bank reference number if any for reconciliation. [To be discussed]

Sample Response – Success

```
{
      "exc": "NSE",
      "sym": "TEST",
      "appNo": "1200299929020",
      "status": 0,
      "sbRefNo": "SB/REF/00022"
}
```

Sample Response – Failed

```
{
      "exc": "NSE",
      "sym": "TEST",
      "appNo": "1200299929020",
      "status": 2,
      "msg": "Symbol not active"
}
```

In case of failures, the exchange user will have option to manually resend the client application request to the sponsor bank, after taking corrective actions if any.

POST /<version>/clientapplicationbulk

This API will allow exchanges to submit one or more IPO application request (new / modification / cancellation) to the sponsor bank.

Method	POST
Request	JSON
Response	JSON

Request JSON

The request JSON structure comprises a list of structures each representing a client application. The structure of the client application is same as "Client Application Request Json" in the Sponsor bank API POST clientapplication.

Sample Request

```
[
      {
            "exc": "NSE",
            "sym": "TEST",
            "appNo": "1200299929020",
            "seqNo": 1,
            "memCd": "99999",
            "cat": "IND",
            "cliNm": "Sumit Aggarwal",
            "dep": "NSDL",
            "dpId": "IN012345",
            "benId": "12345678",
            "pan": "AFAKA2323L",
            "upi": "sumitagg01@upi",
            "entTm" : "01-10-2018 13:40:55",
            "action": "N",
            "bids": [
                  {
                        "bidId": 2018100100000122,
                        "qty": 100,
                        "pr": 55.30,
                        "amt": 5530.00
                  },
                  {
                        "bidId": 2018100100000123,
                        "qty": 106,
                        "pr": 55.00,
                        "amt": 5830.00
                  }
            ]
      },
      {
            "exc": "NSE",
            "sym": "TEST1",
            "appNo": "1200000001",
            "seqNo": 2,
            "memCd": "88888",
            "cat": "IND",
            "cliNm": "Ajay Kumar",
            "dep": "CDSL",
            "dpId": null,
            "benId": "1231231212345678",
            "pan": "AFAKA9999L",
            "upi": "ajaykumardummy@upi",
            "entTm" : "01-10-2018 13:41:05",
            "action": "N",
            "bids": [
                  {
                        "bidId": 2018100100000131,
                        "qty": 200,
                        "pr": 55.30,
                        "amt": 11060.00
                  },
                  {
                        "bidId": 2018100100000132,
                        "qty": 206,
                        "pr": 55.00,
                        "amt": 11330.00
                  }
            ]
      }
]
```

Response JSON

The response JSON structure comprises a list of structures each representing a response status of client application in request json. The structure of the response json is same as "Client Application Response Json" in the Sponsor bank API POST clientapplication.

Sample Response – Success

```
[
      {
            "exc": "NSE",
            "sym": "TEST",
            "appNo": "1200299929020",
            "status": 0,
            "sbRefNo": "SB/REF/00022"
      },
      {
            "exc": "NSE",
            "sym": "TEST1",
            "appNo": "1200000001",
            "status": 2,
            "msg": "Symbol not active"
      }
]
```

POST /<version>/clientapplicationstatus

This API will allow exchange to query the payment status of one or more client application requests. The sponsor bank would respond with entire log of payment status updates for each application number. The API should support querying of a maximum of 200 application numbers in a single call. If request for a particular application number is not available with the sponsor bank then log list for that application should be empty.

Method	POST
Request	JSON
Response	JSON

Request JSON

Field	Type	Mand.	Description
exc	String(3)	Yes	Unique Identifier of Exchange (Same as LoginId in header)
sym	String(10)	Yes	Symbol
appNoList	List<String(16)>	Yes	List of application numbers whose payment status is being queried

Sample Request

```
{
        "exc": "NSE",
        "sym": "TEST",
        "appNoList": [
                "1200299929020",
                "1200299929021",
                "1200299929023"
        ]
}
```

Response JSON

List of client application detail log

Field	Type	Mand.	Description
appNo	String(16)	Yes	Application No.(of the physical form) Symbol and Application No together uniquely identify a client application
logList	List<ClientApplicationStatus>	Yes	List of client application statuses, sorted by "time" in ascending order. In case not a single request for the application number has reached the sponsor bank then the list should be empty.

Client Application Status JSON

Field	Type	Mand.	Description
exc	String(3)	Yes	Unique Identifier of Exchange (Same as LoginId in header)
sym	String(10)	Yes	Symbol
appNo	String(16)	Yes	Application No.(of the physical form) Symbol and Application No together uniquely identify a client application

reqSeqNo	Number(9)	Yes	Sequence number in the message sent by exchange using POST clientapplication api
payStatus	Number	Yes	10=Block/Release Request accepted by Sponsor Bank 11= Block/Release Request rejected by Sponsor Bank due to invalid UPI handle 12= Block/Release Request rejected by Sponsor Bank due to other reasons 13 = Block/Release request rejected due to UPI 2.0 not being supported by investor bank. 21=Block/Release Request rejected by Client Bank 22=Block/Release Request rejected by Client Bank due to Technical Reason 31= Block/Release Request rejected by Client 100=Block Request accepted by Client. Payment successful. 110 = Release Request processed successfully.
time	DateTime	Yes	Timestamp of the action. (Last mile of the action)
ifsc	String(11)	No	IFSC of the client bank account. Valid and Mandatory if payStatus = 100
accNo	String(45)	No	Account number of the client bank account. Valid and Mandatory if payStatus = 100
amtBlocked	Decimal(15, 2)	No	Effective Amount blocked after the request is successfully processed. Valid, Mandatory and greater than zero if payStatus = 100. Valid, Mandatory and zero if payStatus = 110
payRefNo	String(50)	No	Reference number for the payment transaction. Valid and Mandatory if payStatus = 100 or payStatus = 110
rejectReason	String(100)	No	**Rejection Reason Code & Message/Description as received from NPCI. **NB**: **Sponsor Banks to refer NPCI error documents for list of Error code & descriptions and mapping with payStatus code
umn	String(100)	Yes	Unique Mandate no received in response from the SPONSOR bank

Sample Response – Success

```
[
      {
            "appNo" : "1200299929020",
            "logList" : [
                  {
                        "exc": "NSE",
                        "sym": "TEST",
                        "appNo": "1200299929020",
                        "reqSeqNo": 1,
                        "payStatus": 10,
                        "time" : "01-10-2018 13:40:55"",
                        "umn": "TEST1200299929020"

                  },
                  {
                        "exc": "NSE",
                        "sym": "TEST",
                        "appNo": "1200299929020",
                        "reqSeqNo": 1,
                        "payStatus": 100,
                        "time" : "01-10-2018 14:40:55",
                        "ifsc": "ABCD0000001",
                        "accNo": "121212121212121",
                        "amtBlocked":5830.00,
                        "payRefNo": "PR/ABCD/00002322",
                        "umn": "TEST1200299929020"
                  }
            ]
      },
      {
            "appNo" : "1200299929021",
            "logList" : [
                  {
                        "exc": "NSE",
                        "sym": "TEST",
                        "appNo": "1200299929021",
                        "reqSeqNo": 1,
                        "payStatus": 10,
                        "time" : "01-10-2018 13:50:55",
                        "umn": "TEST1200299929021"

                  },
                  {
                        "exc": "NSE",
                        "sym": "TEST",
                        "appNo": "1200299929021",
                        "reqSeqNo": 1,
                        "payStatus": 11,
                        "time" : "01-10-2018 13:51:55",
                        "rejectReason": "Invalid UPI handle",
                        "umn": "TEST1200299929021"

                  }
            ]
      },
      {
            "appNo" : "1200299929023",
            "logList" : [
            ]
      }]
```

POST /<version>/ipomaster

This API will allow exchange to notify sponsor bank on any addition / modification in the IPO master.

Method	POST
Request	JSON
Response	JSON

Request JSON

Field	Type	Mand.	Description
exch	String(3)	Yes	Exchange Code
sym	String(10)	Yes	Symbol
name	String(100)	Yes	Issue Name.
biddingStartDate	Date	Yes	Bidding Start Date
biddingEndDate	Date	Yes	Bidding End Date
paymentAcceptance Time	DateTime	Yes	Cutoff time till which payments will have to be accepted from client. Typically this will be T+1, 12:00 hours. Here T is the bidding end date
lastActionTime	DateTime	Yes	The time when the IPO master was added or last updated.
issueType	String(10)	Yes	Type of the issue. Possible values will be EQUITY / DEBT / REIN

Sample Request

```
{
      "exc": "NSE",
      "sym": "TEST",
      "name": "TEST COMPANIES LIMITED",
      "biddingStartDate": "01-10-2018",
      "biddingEndDate": "04-10-2018",
      "paymentAcceptanceTime": "05-10-2018 13:00:00",
      "lastActionTime": "04-10-2018 14:45:12",
      "issueType": "EQUITY"
}
```

```
{
      "exc": "NSE",
      "sym": "TESTFUT",
      "name": "TEST FUTURE COMPANIES LIMITED",
      "biddingStartDate": "01-10-2018",
      "biddingEndDate": "04-10-2018",
      "paymentAcceptanceTime": "05-10-2018 13:00:00",
      "lastActionTime": "04-10-2018 14:45:12",
      "issueType": "DEBT"
}
```

```
{
      "exc": "NSE",
      "sym": "TESTREIN",
      "name": "TEST REIN COMPANIES LIMITED",
      "biddingStartDate": "01-10-2018",
      "biddingEndDate": "04-10-2018",
      "paymentAcceptanceTime": "05-10-2018 13:00:00",
      "lastActionTime": "04-10-2018 14:45:12",
      "issueType": "REIN"
}
```

Response JSON

Field	Type	Mand.	Description
status	Number	Yes	0 = Success Number greater than 0 = Failed. See section “Message Format” above for error status codes.
msg	String(100)	No	Error message in case of failure.

Sample Response - Success

```
{
      "status": 0
}
```

Sample Response - Failure

```
{
      "status": 3,
      "msg": "Unexpected error in exchange system"
}
```

Exchange API

GET /<version>/ipomaster

This API allows sponsor bank to query all active, forthcoming and closing IPOs. The sponsor bank is expected to query the API at fixed intervals (preferably an hour) and update its masters. The response will contain list of objects, with each object representing an IPO.

Method	GET
Request	NONE
Response	JSON

Sample Request URL

```
GET https://<baseurl>/v1/ipomaster
```

Response JSON

Response JSON will contain list of IPO Detail Structure. The IPO Detail Structure will be same as "IPO Master JSON" in Sponsor Bank API POST ipomaster

Sample Response – Success

```
[
	{
		"exc": "NSE",
		"sym": "TEST",
		"name": "TEST COMPANIES LIMITED",
		"biddingStartDate": "01-10-2018",
		"biddingEndDate": "04-10-2018",
		"paymentAcceptanceTime": "05-10-2018 13:00:00",
		"issueType": "EQUITY"
	},
	{
		"exc": "NSE",
		"sym": "TESTFUT",
		"name": "TEST FUTURE COMPANIES LIMITED",
		"biddingStartDate": "16-10-2018",
		"biddingEndDate": "18-10-2018",
		"paymentAcceptanceTime": "19-10-2018 13:00:00",
		"issueType": "DEBT"
	},
	{
		"exc": "NSE",
		"sym": "TESTREIN",
		"name": "TEST REIN COMPANIES LIMITED",
		"biddingStartDate": "16-10-2018",
		"biddingEndDate": "18-10-2018",
		"paymentAcceptanceTime": "19-10-2018 13:00:00",
		"issueType": "REIN"
	}
]
```

Sample Response - Failure

```
{
	"status": 3,
	"msg": "Unexpected error in exchange system"
}
```

PUT /<version>/clientapplicationstatus

This API will allow the sponsor bank to update payment status of an application.

Method	PUT
Request	JSON
Response	JSON

Request JSON

The request JSON structure is same as "Client Application Status JSON" in the Sponsor bank API POST clientapplicationstatus.

Sample Request

```
{
      "exc": "NSE",
      "sym": "TEST",
      "appNo": "1200299929020",
      "reqSeqNo": 1,
      "payStatus": 100,
      "time" : "01-10-2018 14:40:55",
      "ifsc": "ABCD0000001",
      "accNo": "121212121212121",
      "amtBlocked":5830.00,
      "payRefNo": "PR/ABCD/00002322",
      "umn": "1200299929020"
}
```

Response JSON

Field	Type	Mand.	Description
exc	String(3)	Yes	Unique Identifier of Exchange (Same as LoginId in header)
sym	String(10)	Yes	Symbol
appNo	String(16)	Yes	Application No.(of the physical form) Symbol and Application No together uniquely identify a client application
status	Number	Yes	0 = Success Number greater than 0 = Failed. See section "Message Format" above for error status codes.
msg	String(100)	No	Error message in case of failure.

Sample Response

```
{
      "exc": "NSE",
      "sym": "TEST",
      "appNo": "1200299929020",
      "status": 0
}
```

Sample Response – Failed

```
{
      "exc": "NSE",
      "sym": "TEST",
      "appNo": "1200299929020",
      "status": 2,
      "msg": "Payin not allowed now"
}
```

POST /<version>/clientapplicationstatusbulk

This API will allow the sponsor bank to update payment status of one or more applications.

Method	POST
Request	JSON
Response	JSON

Request JSON

The request JSON structure comprises a list of structures each representing a client application. The structure of the client application is same as "Client Application Status Request JSON" in the Sponsor bank API POST clientapplicationstatus.

Sample Request

```
[
	{
		"exc": "NSE",
		"sym": "TEST",
		"appNo": "1200299929020",
		"reqSeqNo": 1,
		"payStatus": 100,
		"time" : "01-10-2018 14:40:55",
		"ifsc": "ABCD0000001",
		"accNo": "121212121212121",
		"amtBlocked":5830.00,
		"payRefNo": "PR/ABCD/00002322",
		"umn": "1200299929020"
	},
	{
		"exc": "NSE",
		"sym": "TEST",
		"appNo": "120999999",
		"reqSeqNo": 2,
		"payStatus": 10,
		"time" : "01-10-2018 14:41:55"
	}
]
```

Response JSON

The response JSON structure comprises a list of structures each representing a response status for each client application in request json. The structure of the response json is same as "Response JSON" in the exchange API PUT clientapplicationstatus.

Sample Response

```
[
	{
		"exc": "NSE",
		"sym": "TEST",
		"appNo": "1200299929020",
		"status": 0
	},
	{
		"exc": "NSE",
		"sym": "TEST",
		"appNo": "120999999",
		"status": 2,
		"msg": "Payin not allowed now"
```

```
            }
]
```

Sponsor Bank File Interface

This file will have to be generated by sponsor bank after the end of issue. The file will have to be given to the RTA via SFTP (server hosted on exchange premise). The file should contain status of all payment requests initiated by exchange (for new orders, modifications and cancellations) using sponsor bank API **'POST clientapplication'**. The structure and contents for each payment request in the file will be similar to the one in the sponsor bank API **'POST clientapplicationstatus'**.

The purpose of the file is to do reconciliation after end of the issue. Also once reconciliation is complete the exchange will not entertain any update to any payment request for the issue using online API.

Initially while transferring the file, the extension of the file name should be ".tmp" instead of ".csv". Once transfer is complete, the Sponsor bank system will have to rename and change the extension to ".csv" on SFTP server. This mechanism will be required to prevent processing of in-transit files.

End of Issue Reconciliation file (RECO)

File Format	Comma delimited ASCII encoded File With one header record and zero or more data records Field separator comma character (,) Record separator carriage return character followed by new line character (\r\n)
Proposed Naming Convention	RECO_<exc>_<sym>_<sb>_<batchno>.csv where <exc> : Exchange code <sym> : Symbol <sb> : Sponsor bank code <batchno> : 2 digit batch number starting from '01' upto '99'

Header Record Structure

Field	Type	Mand.	Description
exc	String(3)	Yes	Exchange identifier code. Same as in file name.
sym	String(10)	Yes	Symbol. Same as in file name.
sb	String(4)	Yes	Sponsor bank code. Same as in file name.
batchno	Number(2)	Yes	2 digit batch number. Same as in file name.
count	Number	Yes	Count of data records.

Data Record Structure

Field	Type	Mand.	Description
exc	String(3)	Yes	Unique Identifier of Exchange (Same as LoginId in header)
sym	String(10)	Yes	Symbol
appNo	String(16)	Yes	Application No.(of the physical form) Symbol and Application No together uniquely identify a client application
dpId	String(8)	No	DP ID (exactly 8 alphanumeric characters) Ignored in case of CDSL.
benId	String(16)	Yes	Client ID in case of NSDL (exactly 8 alphanumeric characters) Beneficiary ID in case of CDSL (exactly 16 alphanumeric characters)

pan	String(10)	Yes	Permanent Account Number
reqSeqNo	Number(9)	Yes	Sequence number in the message sent by exchange using POST clientapplication api
payStatus	Number	Yes	10=Block/Release Request accepted by Sponsor Bank 11= Block/Release Request rejected by Sponsor Bank due to invalid UPI handle 12= Block/Release Request rejected by Sponsor Bank due to other reasons 13 = Block/Release request rejected due to UPI 2.0 not being supported by investor bank. 21=Block/Release Request rejected by Client Bank 22=Block/Release Request rejected by Client Bank due to Technical Reason 31= Block/Release Request rejected by Client 100=Block Request accepted by Client. Payment successful. 110 = Release Request processed successfully.
time	DateTime	Yes	Timestamp of the action. (Last mile of the action)
ifsc	String(11)	No	IFSC of the client bank account. Valid and Mandatory if payStatus = 100
accNo	String(45)	No	Account number of the client bank account. Valid and Mandatory if payStatus = 100
amtBlocked	Decimal(15, 2)	No	Effective Amount blocked after the request is successfully processed. Valid, Mandatory and greater than zero if payStatus = 100. Valid, Mandatory and zero if payStatus = 110
payRefNo	String(50)	No	Reference number for the payment transaction. Valid and Mandatory if payStatus = 100 or payStatus = 110
rejectReason	String(100)	No	**Rejection Reason Code & Message/Description as received from NPCI. **NB**: **Sponsor Banks to refer NPCI error documents for list of Error code & descriptions and mapping with payStatus code
umn	String(100)	Yes	Unique Mandate no received in response from the SPONSOR bank

Sponsor Bank – RNU File

This file will have to be generated by sponsor banks and has to share to the respective RTA

File Format	Comma delimited ASCII encoded File With one header record and zero or more data records Field separator comma character (,) Record separator carriage return character followed by new line character (\r\n)
Proposed Naming Convention	RNU_<exc>_<sym>_<sb>_<batchno>.zip where <exc> : Exchange code <sym> : Symbol <sb> : Sponsor bank code <batchno> : 2 digit batch number

Header Record Structure

Field	Type	Mand.	Description
exc	String(3)	Yes	Exchange identifier code. Same as in file name.
sym	String(10)	Yes	Symbol. Same as in file name.
rta	String(10)	Yes	RTA Code.
sb	String(4)	Yes	Sponsor bank code. Same as in file name.
batchno	Number(2)	Yes	2 digit batch number. Same as in file name.
count	Number	Yes	Count of data records.

Data Record Structure

Field	Type	Mand.	Description
sym	String(10)	Yes	Symbol
userId	String(30)	Yes	Exchange code for entity placing the bid
appNo	String(16)	Yes	Application No.(of the physical form) Symbol and Application No together uniquely identify a client application
dpId	String(8)	No	DP ID (exactly 8 alphanumeric characters) Ignored in case of CDSL.
benId	String(16)	Yes	Client ID in case of NSDL (exactly 8 alphanumeric characters) Beneficiary ID in case of CDSL (exactly 16 alphanumeric characters)
qty	Number	Yes	Bid Quantity
pr	Decimal(10, 2)	Yes	Bid Price
amt	Decimal(15, 2)	Yes	Bid Amount.
pan	String(10)	Yes	Permanent Account Number
cat	String(5)	Yes	Category of the Investor
entTm	DateTime	Yes	Application entry time in the Exchange Bidding System
modTm	DateTime	No	Last modification time in the Exchange Bidding System
bidId	Number(16)	Yes	Unique Identifier for the allocated bid.
upi	String(45)	Yes	Universal payment identifier of the client
amtBlocked	Decimal(15, 2)	Yes	Amount blocked by sponsor bank
ifsc	String(11)	Yes	IFSC of the client bank account.
accNo	String(45)	Yes	Account number of the client bank account.
payRefNo	String(50)	Yes	Reference number for the payment transaction.
time	DateTime	Yes	Amount block confirm timestamp
allocatedQty	Number	Yes	Quantity allocated. Can be zero.
amtTransfer	Decimal(15, 2)	Yes	Amount to be debited from client account. Can be zero.
amtRefund	Decimal(15, 2)	Yes	Amount to be refunded back (block release) to client account. Can be zero.
reason	String(100)	No	Allocation remarks, if any
umn	String(100)	Yes	Unique Mandate no received in response from the SPONSOR bank
sbStatus	Number	Yes	Status of fund transfer / refund request 0= Instruction executed successfully

			Any Number greater than zero= Failure in execution. The number would indicate error code.
sbRef	String(50)	No	Sponsor Bank Reference Number, if any for execution of the transfer / refund instruction.
sbReason	String(100)	No	Reason in case of failure of request.

Sponsor Bank – Debit Unblock File

This file will have to be generated by sponsor banks and has to share to the respective RTA

File Format	Comma delimited ASCII encoded File With one header record and zero or more data records Field separator comma character (,) Record separator carriage return character followed by new line character (\r\n)
Proposed Naming Convention	DBT_<exc>_<sym>_<sb>_<batchno>.zip where <exc> : Exchange code <sym> : Symbol <sb> : Sponsor bank code <batchno> : 2 digit batch number

Header Record Structure

Field	Type	Mand.	Description
exc	String(3)	Yes	Exchange identifier code. Same as in file name.
sym	String(10)	Yes	Symbol. Same as in file name.
sb	String(10)	Yes	RTA Code.
batchno	Number(2)	Yes	2 digit batch number. Same as in file name.
count	Number	Yes	Count of data records.

Data Record Structure

Field	Type	Mand.	Description
sym	String(10)	Yes	Symbol
userId	String(30)	Yes	Exchange code for entity placing the bid
appNo	String(16)	Yes	Application No.(of the physical form) Symbol and Application No together uniquely identify a client application
dpId	String(8)	No	DP ID (exactly 8 alphanumeric characters) Ignored in case of CDSL.
benId	String(16)	Yes	Client ID in case of NSDL (exactly 8 alphanumeric characters) Beneficiary ID in case of CDSL (exactly 16 alphanumeric characters)
qty	Number	Yes	Bid Quantity
pr	Decimal(10, 2)	Yes	Bid Price
amt	Decimal(15, 2)	Yes	Bid Amount.
pan	String(10)	Yes	Permanent Account Number
cat	String(5)	Yes	Category of the Investor

entTm	DateTime	Yes	Application entry time in the Exchange Bidding System
modTm	DateTime	No	Last modification time in the Exchange Bidding System
bidId	Number(16)	Yes	Unique Identifier for the allocated bid.
upi	String(45)	Yes	Universal payment identifier of the client
amtBlocked	Decimal(15, 2)	Yes	Amount blocked by sponsor bank
ifsc	String(11)	Yes	IFSC of the client bank account.
accNo	String(45)	Yes	Account number of the client bank account.
payRefNo	String(50)	Yes	Reference number for the payment transaction.
time	DateTime	Yes	Amount block confirm timestamp
allocatedQty	Number	Yes	Quantity allocated. Can be zero.
amtTransfer	Decimal(15, 2)	Yes	Amount to be debited from client account. Can be zero.
amtRefund	Decimal(15, 2)	Yes	Amount to be refunded back (block release) to client account. Can be zero.
reason	String(100)	No	Allocation remarks, if any
umn	String(100)	Yes	Unique Mandate no received in response from the SPONSOR bank
sbStatus	Number	Yes	Status of fund transfer / refund request 0= Instruction executed successfully Any Number greater than zero= Failure in execution. The number would indicate error code.
sbRef	String(50)	No	Sponsor Bank Reference Number, if any for execution of the transfer / refund instruction.
sbReason	String(100)	No	Reason in case of failure of request.

RTA Interface

Process

- The RTA would send a final fund transfer/release file to the sponsor bank using SFTP server hosted by the exchange. The file will be placed in home directory of the RTA. Initially while transferring the file, the extension of the file name should be ".tmp" instead of ".zip". Once transfer is complete, the RTA system will have to rename and change the extension to ".zip" on SFTP server. This mechanism will be required to prevent processing of in-transit files.
- Once file with extension ".zip is detected, the exchange system will move the file to home directory of respective sponsor bank on the SFTP server. Exchange system will only validate the name of the file. Exchange system will not read or validate the contents of the file.
- The sponsor bank would pick the file and move the file to "archive" directory available in the home directory of the sponsor bank on SFTP server.
- After processing the file, the sponsor bank would put response file in its home directory. Mechanism similar to one used by RTA to prevent processing of in-transit files will have to be employed by sponsor bank as well.
- The exchange system will move the response file to the home directory of RTA.
- The RTA would pick the response file and move the same to "archive" directory available in its home directory.
- The response file will then be processed by RTA system.

RTA Fund Transfer / Release file

File Format	Comma delimited ASCII encoded File With one header record and zero or more data records Field separator comma character (,) Record separator carriage return character followed by new line character (\r\n)
Proposed Naming Convention	FND_<exc>_<sym>_<rta>_<sb>_<batchno>.zip where <exc> : Exchange code <sym> : Symbol <rta> : RTA Code <sb> : Sponsor bank code <batchno> : 2 digit batch number starting from '01' upto '99'

Header Record Structure

Field	Type	Mand.	Description
exc	String(3)	Yes	Exchange identifier code. Same as in file name.
sym	String(10)	Yes	Symbol. Same as in file name.
rta	String(10)	Yes	RTA Code. Same as in file name.
sb	String(4)	Yes	Sponsor bank code. Same as in file name.
batchno	Number(2)	Yes	2 digit batch number. Same as in file name.
count	Number	Yes	Count of data records.

Data Record Structure

Field	Type	Mand.	Description
sym	String(10)	Yes	Symbol
userId	String(30)	Yes	Exchange code for entity placing the bid
appNo	String(16)	Yes	Application No.(of the physical form) Symbol and Application No together uniquely identify a client application
dpId	String(8)	No	DP ID (exactly 8 alphanumeric characters) Ignored in case of CDSL.
benId	String(16)	Yes	Client ID in case of NSDL (exactly 8 alphanumeric characters) Beneficiary ID in case of CDSL (exactly 16 alphanumeric characters)
qty	Number	Yes	Bid Quantity
pr	Decimal(10, 2)	Yes	Bid Price
amt	Decimal(15, 2)	Yes	Bid Amount.
pan	String(10)	Yes	Permanent Account Number
cat	String(5)	Yes	Category of the Investor
entTm	DateTime	Yes	Application entry time in the Exchange Bidding System
modTm	DateTime	No	Last modification time in the Exchange Bidding System
bidId	Number(16)	Yes	Unique Identifier for the allocated bid.
upi	String(45)	Yes	Universal payment identifier of the client
amtBlocked	Decimal(15, 2)	Yes	Amount blocked by sponsor bank
ifsc	String(11)	Yes	IFSC of the client bank account.
accNo	String(45)	Yes	Account number of the client bank account.
payRefNo	String(50)	Yes	Reference number for the payment transaction.
time	DateTime	Yes	Amount block confirm timestamp
allocatedQty	Number	Yes	Quantity allocated. Can be zero.
amtTransfer	Decimal(15, 2)	Yes	Amount to be debited from client account. Can be zero.
amtRefund	Decimal(15, 2)	Yes	Amount to be refunded back (block release) to client account. Can be zero.
reason	String(100)	No	Allocation remarks, if any
umn	String(100)	Yes	Unique Mandate no received in response from the SPONSOR bank

RTA Fund Transfer / Release Response file

File Format	Same as RTA Fund Transfer/Release file
Proposed Naming Convention	FNDRESP_<exc>_<sym>_<rta>_<sb>_<batchno>.zip Place holders will have same values as in the "RTA Fund Transfer/Release file".

Header Record Structure

Same as the header record structure of "RTA Fund Transfer/Release file"

Data Record Structure

Same as the data record structure of "RTA Fund Transfer/Release file" with following fields appended at the end of each data record

Field	Type	Mand.	Description
sbStatus	Number	Yes	Status of fund transfer/ refund request 0=Instruction executed successfully Any number greater than zero=Failure in instruction execution. The number would indicate error code.
sbRef	String(50)	No	Sponsor Bank Reference number, if any for execution of the transfer/refund instruction.
sbReason	String(100)	No	Reason in case of failure of request.

Allotment file

File Format	Comma delimited ASCII encoded File With one header record and zero or more data records Field separator comma character (,) Record separator carriage return character followed by new line character (\r\n)
Proposed Naming Convention	<EXC>_<SYM>_ALLOTMENT_< ddMMyyyy >_<RET>.CSV where < EXC > : Exchange code(NSE) <SYM>: Symbol

Header Record Structure

Field	Type	Mand.	Description
recordType	Number	Yes	Record Type (default 10)
recordCount	Number	Yes	Count of data records.
totalAllotmentQuantity	Number	Yes	Sum of allotment quantity in file

Data Record Structure

Field	Type	Mand.	Description
recordType	Number(2)	Yes	Record Type (default 20)
sym	String(10)	Yes	Symbol
appNo	String(16)	Yes	Application No Symbol and Application No together uniquely identify a client application
dpId	String(8)	No	DP ID (exactly 8 alphanumeric characters) Ignored in case of CDSL.
benId	String(16)	Yes	Client ID in case of NSDL (exactly 8 alphanumeric characters) Beneficiary ID in case of CDSL (exactly 16 alphanumeric characters)
pan	String(10)	Yes	Permanent Account Number
orderNumber	Number (16)	Yes	Unique Identifier for the allocated bid.
allotmentQty	Number (11)	Yes	Allotment Quantity
allotmentPrice	Number (20)	Yes	Allotment Price
technicalRejection Records	String(3)	Yes	Technical rejection records

Exchange-Sponsor Bank Reco File Interface

This file will have to be generated by exchange on daily basis at EOD till issue closure. The file will have to be given to the Sponsor Bank via SFTP (server hosted on exchange premise). The file should contain status of all payment requests initiated by exchange (for new orders, modifications and cancellations) using sponsor bank API **'POST clientapplication'**. The structure and contents for each payment request in the file will be similar to the one in the sponsor bank API **'POST clientapplicationstatus'**.

The purpose of this file for sponsor bank is to do reconciliation on daily basis at sponsor bank end.

Initially while transferring the file, the extension of the file name should be ".tmp" instead of ".csv". Once transfer is complete, the Exchange system will have to rename and change the extension to ".csv" on SFTP server. This mechanism will be required to prevent processing of in-transit files.

Daily Issue Reconciliation file (RECO)

File Format	Comma delimited ASCII encoded File With one header record and zero or more data records Field separator comma character (,) Record separator carriage return character followed by new line character (\r\n)
Proposed Naming Convention	EXCHRECO_<exc>_<sym>_<sb>_<ddmmyyyyhhmmss>_<batchno>.csv where <exc> : Exchange code <sym> : Symbol <sb> : Sponsor bank code < ddmmyyyyhhmmss >:Indicates Datetime stamp of file generation <batchno> : 2 digit batch number

Header Record Structure

Field	Type	Mand.	Description
exc	String(3)	Yes	Exchange identifier code. Same as in file name.
sym	String(10)	Yes	Symbol. Same as in file name.
sb	String(4)	Yes	Sponsor bank code. Same as in file name.
batchno	Number(2)	Yes	2 digit batch number. Same as in file name.
count	Number	Yes	Count of data records.

Data Record Structure

Field	Type	Mand.	Description
exc	String(3)	Yes	Unique Identifier of Exchange (Same as LoginId in header)
sym	String(10)	Yes	Symbol
appNo	String(16)	Yes	Application No.(of the physical form) Symbol and Application No together uniquely identify a client application
dpId	String(8)	No	DP ID (exactly 8 alphanumeric characters) Ignored in case of CDSL.
benId	String(16)	Yes	Client ID in case of NSDL (exactly 8 alphanumeric characters) Beneficiary ID in case of CDSL (exactly 16 alphanumeric characters)
pan	String(10)	Yes	Permanent Account Number
reqSeqNo	Number(9)	Yes	Sequence number in the message sent by exchange using POST clientapplication api

payStatus	Number	Yes	10=Block/Release Request accepted by Sponsor Bank 11= Block/Release Request rejected by Sponsor Bank due to invalid UPI handle 12= Block/Release Request rejected by Sponsor Bank due to other reasons 13 = Block/Release request rejected due to UPI 2.0 not being supported by investor bank. 21=Block/Release Request rejected by Client Bank 22=Block/Release Request rejected by Client Bank due to Technical Reason 31= Block/Release Request rejected by Client 100=Block Request accepted by Client. Payment successful. 110 = Release Request processed successfully.
time	DateTime	Yes	Timestamp of the action. (Last mile of the action)
ifsc	String(11)	No	IFSC of the client bank account. Valid and Mandatory if payStatus = 100
accNo	String(45)	No	Account number of the client bank account. Valid and Mandatory if payStatus = 100
amtBlocked	Decimal(15, 2)	No	Effective Amount blocked after the request is successfully processed. Valid, Mandatory and greater than zero if payStatus = 100. Valid, Mandatory and zero if payStatus = 110
payRefNo	String(50)	No	Reference number for the payment transaction. Valid and Mandatory if payStatus = 100 or payStatus = 110
rejectReason	String(100)	No	**Rejection Reason Code & Message/Description as received from NPCI. **NB**: **Sponsor Banks to refer NPCI error documents for list of Error code & descriptions and mapping with payStatus code
umn	String(100)	Yes	Unique Mandate no received in response from the SPONSOR bank
bidId	Number(16)	Yes	Unique Identifier for the bid.
deleteflag	Char(1)	Yes	Bid Active Status. 'Y' – Deleted/Canceled Record. 'N' – Active Record

Note: This file will have only one record for an **application number + bid** id of the IPO with the latest pay status available at exchange end.

Closed User Group (CUG)

This file will have to be generated by exchange and will be send to respective sponsor banks via Sftp

File Format	Comma delimited ASCII encoded File With one header record and zero or more data records Field separator comma character (,) Record separator carriage return character followed by new line character (\r\n)
Proposed Naming Convention	<exc>_CUG_<ddMMyyyy>.zip where <exc> : Exchange code

Header Record
Member Code, Member Name, Email Id

Data Record Structure

Field	Type	Mand.	Description
Member Code	String(10)	Yes	Member Code
Member Name	String(30)	Yes	Member Name
Email Id	String(60)	Yes	Email ID

SFTP File Path

SFTP Path	Description
<ENTITY-CODE>/inbox	The file will be received and consumed by the registered entity
<ENTITY-CODE>/send	The file will be placed by the source entity and will be transferred to the inbox directory of the receipt entity.