Technical Documentation E-COLLECTION API

Product Brief: eCollection is a product for making virtual account number based payments. eCollection API is used for sending e-collection MIS to client via API post service. The MIS will have standard fields, which are available in e-collection MIS for the modes of collections as IMPS, RTGS and NEFT.

 Contents

Note:-

1. Hybrid Encryption supports both 2048- & 4096-bit certificate
2. Only public certificate is required for Ecollection Integration and SSL is not required.

A. Prerequisites:

✓ HTTPS back End URL for both LIVE and UAT
✓ Public certificate in 2048 or 4096 bit in .cer or .txt (Public cert must be CA certified in production)
✓ IP and port for both LIVE and UAT (Mandatory to share if URL is IP based)
✓ Account Details- Client Code & Client Name

B. Integration steps:

✓ Client code needs to be created under Profunds. It can be 3, 4, or 6-digit client code.
✓ Whitelisting of connecting configuration (IP & Port) at both end.
✓ Handshake, Encryption Logic implementation and API integration to be completed
✓ Testing under UAT
✓ UAT sign off and then live deployment

C. Encryption Logic – Hybrid encryption

Encryption logic:

Hybrid Encryption & Decryption Process

Sample Encrypted Request to be sent to Client (JSON):

{

"requestId": "",

"service": "LOP",

"encryptedKey":
"oG5mU1JJNBuwQaSLKb3wfRZks/cT2Vo2yBNNuqjNHDWEC144WxC8iKqBpJAgq7reFKC4
sHNUmNPRDya1AvmQ7x1L+3EAdEs9FEWNurZuWTvZpk4y7JrGhg0rz9KptBf+JfJUkSMo7N
R3Saxel6EYtckkDr3AGW7WJZmhcEoAMMXRws/hLVmaNHC/nOjCNqqBd4lOOAzdJh/HAD
RVI+YAJKT8dE4x9NTl+UX1zAooWhza+TsWEHfxzQla7zai7WSa/wiJD3uD7mk5vT1WY/fKJ
BquCuzM7l35vigDhmb7dLVLuX8VMiNQrtErWNl0uVaag1jg+uZUtyDSxjPFi5yEpKVVc7+T5
03lDnCvkCFDygqasDsPL24qOjYk4XavTZvwGuPAdYNNkVnLzVElEhg4zS2ye+fa/8fZiMt/3fw
YeN9dgn9i5R6VOFbXSuZJYPSci9k0oqz73h1nzFtps60rUEDoGlkGvm9waJU3W78VH5mIdG
fGvvJjiKluVHmi/huzEX9v4w3mW7RDGgmOuKlmkqki+XWgyB0JvVmsLdO+cBaym/seZP3+z
dfhO9AWSI2tDLD4Vf0jDjzoDSFN2mzUFgHK9mbtbXgvsnReoGqx/KsivzmZNLmDmtg8eR4
Z9LnLni4rl4OtkDv5y/mxMtL3MBUUUajkw6OS6NnhEG895yo=",

"oaepHashingAlgorithm": "NONE",

"iv": "",

"encryptedData":
"wBJSefFsnJVlobh1cJR553w6Ay6b8/2frCjxvdZ1Bsnxztsul7Ha8lFl4PoZD+lhdlRShWdKgz3y
JYlisGV/KKpyMSY3DILOpbkqEa0Qq0g=",

"clientInfo": "",

"optionalParam": ""

}

Field Description:

Field	Description	Data Type	Max length	Mandatory	Permitted Values
requestId	Unique identifier for the request. Not being stored at any level.	String	64	No	

service type	Service Name ; Identifying the backend service name	String		Yes	
encryptedKey	One-time use AES key encrypted by the Client's public key. Requirement is for a 128-bit key (with 256-bit key supported as an option).	String. Base64 - encoded data (case-insensitive)		Yes	
oaepHashingAlgorithm	Describes the algorithm used for Asymmetric Encryption of one time AES key.	String	6	Yes	Value : NONE, Meaning : RSA/ECB/PKCS1 is used for Asymmetric Encryption Value : SHA1, Meaning : RSA/NONE/OAEPWithSHA1AndMGF1Padding OR RSA/ECB/OAEPWithSHA1AndMGF1Padding

iv	The initialization vector used when encrypting data using the one-time use AES key.	String. Base64 - encoded data (case-insensitive)	24	Yes, if IV is not part of encrypted data itself. Leave blank otherwise. For the response data encrypted at ICICI end, IV will always be part of encryptedData itself and it is randomly generated for each request. Can be retrieved by Client by retrieving the first 16 bytes of Base64 decoded encryptedData	Exactly 16 bytes actual value to match the block size
encryptedData	Contains the encrypted dataPayload object containing the business information. Encrypted by the ephemeral AES key using AES/CBC/PKCS5 Padding. Sample unencrypted object: Please refer first section of document for a sample object.	String. Base64 - encoded data (case-insensitive)		Yes	

clientInfo	ClientIP or other information	String		No	
optionalParam	Reserved for future use	String		No	

For encryption of request at ICICI:

SesionKey = Randomly generated string of length 16 (OR 32).

encryptedKey = Base64Encode(RSA/ECB/PKCS1Encryption(SesionKey,ICICIPubKey.cer))

IV = Initialization Vector- Exactly 16 bytes actual value to match the block size

encryptedData = Base64Encode(AES/CBC/PKCS5Padding(Request,SessionKey, IV))

For encryption of response at ICICI:

encryptedKey = Base64Encode(RSA/ECB/PKCS1Encryption(SesionKey,ClientPubKey.cer))

Session key is nothing but randomly generated string of length 16 (OR 32).

encryptedData = Base64Encode(AES/CBC/PKCS5Padding(Response,SessionKey))

For decryption of response at Client:

IV= getFirst16Bytes(Base64Decode(encryptedData)

SessionKey = Base64Decode(RSA/ECB/PKCS1Decryption(encryptedKey,ClientPrivateKey.p12,))

Session key is nothing but randomly generated string of length 16 (OR 32) .

Response = Base64Decode (AES/CBC/PKCS5Padding Decryption(encryptedData,SessionKey, IV))

Or

Steps for Encryption

1) Generate 16 digit random number. Say RANDOMNO.
2) Encrypt RANDOMNO using RSA/ECB/PKCS1Padding and encode using Base64. Say encryptedKey.
3) Perform AES/CBC/PKCS5Padding encryption on request payload using RANDOMNO as key and iv- initialization vector. Say encryptedData.
4) Now client may choose to send IV in request from one of below two options.
    a. Send Base64 Encoded IV in “iv” tag. (Recommended Approach)
    b. Send IV as a part of encryptedData itself.

        bytes[] iv = IV;

        bytes[] cipherText = symmetrically encrypted Bytes (step3)

        bytes[] concatB = iv + cipherText;

        encryptedData = B64Encode(concatB);

5) Perform AES/CBC/PKCS5Padding encryption on DATA using RANDOMNO as key and Base64encoded RANDOMNO as IV. Say ENCR_DATA.

Steps for Decryption

1) Get the IV- Base64 decode the encryptedData and get first 16 bytes and rest is encrypted response.
    bytes[] IV= getFirst16Bytes(Base64Decode(encryptedData)
2) Decrypt encryptedKey using algo (RSA/ECB/PKCS1Padding) and Client’s private key.
3) Decrypt the response using algo (AES/CBC/PKCS5Padding).
4) Ignore first 16 bytes of response, as it contains IV.

cURL command

```
curl -X POST \
 'https://apigwuat.icicibank.com:8443/<relativeURI>' \
 -H 'apikey: <apikey> \
 -H 'content-type: application/json' \
```

```
 -d '{
   "requestId": "",
   "service": "",
   "encryptedKey":
"oG5mU1JJNBuwQaSLKb3wfRZks/cT2Vo2yBNNuqjNHDWEC144WxC8iKqBpJAgq7reFKC4
sHNUmNPRDya1AvmQ7x1L+3EAdEs9FEWNurZuWTvZpk4y7JrGhg0rz9KptBf+JfJUkSMo7N
R3Saxel6EYtckkDr3AGW7WJZmhcEoAMMXRws/hLVmaNHC/nOjCNqqBd4IOOAzdJh/HAD
RVI+YAJKT8dE4x9NTl+UX1zAooWhza+TsWEHfxzQIa7zai7WSa/wiJD3uD7mk5vT1WY/fKJ
BquCuzM7l35vigDhmb7dLVLuX8VMiNQrtErWNI0uVaag1jg+uZUtyDSxjPFi5yEpKVVc7+T5
03lDnCvkCFDygqasDsPL24qOjYk4XavTZvwGuPAdYNNkVnLzVElEhg4zS2ye+fa/8fZiMt/3fw
YeN9dgn9i5R6VOFbXSuZJYPSci9k0oqz73h1nzFtps60rUEDoGlkGvm9waJU3W78VH5mIdG
fGvvJjiKluVHmi/huzEX9v4w3mW7RDGgmOuKlmkqki+XWgyB0JvVmsLdO+cBaym/seZP3+z
dfhO9AWSI2tDLD4Vf0jDjzoDSFN2mzUFgHK9mbtbXgvsnReoGqx/KsivzmZNLmDmtg8eR4
Z9LnLni4rl4OtkDv5y/mxMtL3MBUUUajkw6OS6NnhEG895yo=",
   "oaepHashingAlgorithm": "NONE",
   "iv": "",
   "encryptedData":
"wBJSefFsnJVlobh1cJR553w6Ay6b8/2frCjxvdZ1BsnxztsuI7Ha8lFl4PoZD+lhdlRShWdKgz3y
JYlisGV/KKpyMSY3DILOpbkqEa0Qq0g=",
   "clientInfo": "",
   "optionalParam": ""
}'
```

**** In case customer do not want encrypted MIS, we can send the non-encrypted MIS by masking the payer and customer account number fields in the MIS packet.**

D. JSON API specifications - Request and Response Tags

Standard Request and Response:

```
{
```

"ClientCode": "TARF",

"VirtualAccountNumber": "TARF1122",

"Mode": "NEFT",

"UTR": "HDFC174479080102",

"SenderRemark":"testing success1"

"ClientAccountNo": "000405100456",

"Amount": "12",

"PayerName": "Abhay new",

"PayerAccNumber": “0000000009123",

"PayerBankIFSC": "HDFC0000001",

"PayerPaymentDate": "08/01/2020",

"BankInternalTransactionNumber": "N20182294870"

}

Input Parameters

Name	Type	Description	Mandatory (Y/N)
ClientCode	VARCHAR2	Client code	Y
VirtualAccountNumber	VARCHAR2	Virtual account number	Y
Mode	VARCHAR2	Transaction mode	Y
UTR	VARCHAR2	Unique Transaction Reference	Y
SenderRemark	VARCHAR2	Sender Remark	Y
ClientAccountNo	VARCHAR2	Customer Account number	Y
Amount	VARCHAR2	Amount	Y
PayerName	VARCHAR2	Payer name	Y
PayerAccNumber	VARCHAR2	Payer account number	Y
PayerBankIFSC	VARCHAR2	Payer bank IFSC	Y
PayerPaymentDate	VARCHAR2	Payer payment date	Y
BankInternalTransactionNumber	VARCHAR2	Bank Internal Transaction Number	Y

{

"Response":"Success",

"Code":"11"

}

{

"Response": "Duplicate UTR",

"Code": "06"

}

Name	Type	Description	Mandatory (Y/N)
Response	VARCHAR2	Success or Duplicate UTR	Y
Code	VARCHAR2	Response Code	Y

Code	Response
11	Success
06	Duplicate UTR

If there will not be a successful response for a transaction, then system will retrigger the API request until successful response up to 4 reattempts.

Note:-

For the Fund Transfer cases (Ecollection) transactions are without UTR no's for that case you have to validate same using "Bank internal transaction number”. You need to build duplicate UTR logic on "Bank internal transaction number".