2025

# Beneficiary Satyapan Application

Secure-Data-Transfer API Specification Document

*(Developer Usage only)*

VERSION 1.0.1

NATIONAL INFORMATICS CENTRE

A-BLCOK CGO COMPLEX | New Delhi - 110003

***Contact Us:***

**National Informatics Centre**
**# A – Block CGO Complex, Lodhi Road, New Delhi - 110003**

## Contributors:

| Activity | Name of contributor | Designation of contributor |
|---|---|---|
| **Prepared by** | | |
| **Reviewed by** | | |
| **Approved by** | | |

## Amendment log:

| Version | Date | Brief description | Section Change |
|---|---|---|---|
| **1.0.0** | **10th Jun, 2025** | **Initial Doc** | |
| | | | |

## INDEX

# 1. Beneficiary Satyapan Application (BSA)
(An Aadhaar Authentication Platform for Government Schemes)

## 1.1. Introduction

The Beneficiary Satyapan Application (BSA) is a unified, Aadhaar-enabled eKYC authentication platform developed by NIC to streamline beneficiary verification for central and state government schemes. It functions under the NIC AUA/ASA framework and supports various biometric modalities including fingerprint, iris, and face authentication.

The approach described in this document is for securely sharing eKYC data to the on-boarded agency via API with a **hybrid encryption** method. This combines the strengths of both symmetric and asymmetric encryption to ensure robust security, efficiency, and confidentiality during data transfer.

## 1.2. Data Encryption (Symmetric Key Encryption)

The actual eKYC data is encrypted using a symmetric encryption algorithm (AES-256-CBC) to protect the confidentiality of the eKYC data itself. Symmetric key encryption is particularly suitable for encrypting large amounts of data due to its minimal computational overhead. In this method, the same symmetric key used for encryption must also be available to the receiving party for decryption, which poses a challenge for secure key exchange. Ensuring that the symmetric key is securely shared between parties is crucial to maintaining data security.

## 1.3. Key Encryption (Asymmetric Key Encryption)

To securely share the symmetric key used for data encryption, asymmetric key encryption is employed. The symmetric key is encrypted using the public key of the receiving agency (on-boarded agency). This approach guarantees that only the receiving agency, which holds the corresponding private key, can decrypt the symmetric key, thereby gaining access to the actual eKYC data. The RSA algorithm is used for this key encryption process, providing a robust mechanism to secure the symmetric key during transmission.

## 1.4. Data Transfer Workflow

The data transfer workflow begins with the encryption of the user's eKYC data using a symmetric key. Once encrypted, the symmetric key itself is encrypted using the public key of the intended recipient. The encrypted eKYC data and the encrypted symmetric key are then transmitted together via the API. To ensure the security of this transmission, the IP address of the data-requesting entity must be whitelisted at the source end. Typically, the data transfer occurs over a secure channel to prevent man-in-the-middle attacks, safeguarding the integrity and confidentiality of the data during transit.

## 1.5. Decryption at the Receiving End

At the receiving end, the receiving agency first decrypts the symmetric key using its private key. Once the symmetric key is successfully decrypted, it is then used to decrypt the actual eKYC data, thereby restoring it to its original form. This two-step decryption process ensures that only the authorized recipient can access the original eKYC data, maintaining the confidentiality and security of sensitive information throughout the transmission and decryption process.

## 2. API Request and Response.

### 2.1. Generate JWT Token

#### REST API Headers

| Field Name | Description | Value |
|---|---|---|
| Authorization | Not applicable | Not applicable |
| Content-Type | | application/json |

| 1 . API End Point |
|---|
| https://staging.bsa.nic.in/bsadata_api/auth/login |

**Input Parameters:**

| Parameter name | Data Type (Max Length) | Is Mandatory | Description |
|---|---|---|---|
| **scode** | String(2) | Yes | Provided by BSA |
| **service_code** | String(10) | Yes | Provided by BSA |
| **enc_login** | String (33) | Yes | Username/password will be shared by BSA. Encrypted {“username”:”socialsp<YmdHis>”,”password” :”nsap@!321<YmdHis>”} |

**How to Encrypt above string**

**Sample Json String:**
{“username”:”**socialsp**20250710140359”,”password”:”**nsap@!321**20250710140359”}

Json string Encrypted using **KEY and IV**
**KEY , IV , username, password** shared with BSA and Service Agency

Key: D3H56E89G52H5D678H95F3C5E78D9H39
**IV:** 3916547354863579
Encryption mode: AES-256-CBC
Note:
1. After encryption the string to be base64encoded.

| |
|---|
| 2. Bearer token validity may be defined as 24 HRS. |

| **Sample Request** |
|---|
| {"scode":"6","service_code":"HRY001","enc_login":"zzzgtICdSupHfoVwQ"} |
| |
| **Sample Response** |
| {"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQiOiJtaGlhcGkiLCJpYXQiOjE3NTU2NjMxNDMsImV4cCI6MTc1NTY4MTE0M30.ExWMsfJzpSybTz4T71iUqV-Y6UBkCGo4ril_WH-EmPA","status":1} |
| **Sample Response Failure ( json )** |
| If username/password wrong<br>{"status":"2","message":"Invalid Credential"}<br>If not a valid encryption string/request time<br>{"status":"3","message":"Request time out or invalid encryption"} |

***Note – enc_login Encryption step**

1. Concatenation of username:password:YmdHis
2. After concatenation encrypt using KEY and IV.
3. Then base64 encoded.

## 2.2. Pull Beneficiary Data

| **2. API End Point** |
|---|
| https://staging.bsa.nic.in/bsadata_api/bsadata/get_beneficiery_data |

### REST API Headers

| Header | Value |
|---|---|
| Authorization | Authorization: Bearer eyJhbG****JIUzUxMi… |
| Content-Type | application/json |

**Input Parameters:**

| Parameter name | Data Type (Max Length) | Is Mandatory | Description |
|---|---|---|---|
| **username** | String(30) | Yes | Provided by BSA |
| **dlc_creation_date** | String(30) | Yes | Beneficiary Authentication date (YYYY-MM-DD) |
| **service_code** | String(10) | Yes | Provided by BSA |
| **scode** | String(2) | Yes | Provided by BSA |
| **api_txn** | String (33) | Yes | Unique transaction-<br>Ex. NIC-HRY001-20250514-102008-876625<br>NIC-<Service Code>-<YYYYMMDD>-<HHMMSS>-<6 Digit Random Number> |

| **Sample Request ( json )** |
|---|
| {"api_txn":"NIC-HRY001-20250514-101510-123456", "username":" hrydataapi", "dlc_creation_date":"2025-05-06", "service_code":"HRY001","scode":"6"} |

**Sample Response Success ( json )**

{"status": "1", "api_txn": "NIC-HRY001-20250514-102008-876625","dlc_creation_date": "2025-05-06", "enc_data": "aJ0aUeEGf5ZQRCPAxkIp38NaOJUfrwg3g9KJN7wtpgj9j=","enc_key":"QaVib05XoQXHdYXKq7mNuVUHW+7D+bULFIagrzvOR0gkEFtQ0FOqJwGXdQk6ZWuEmU0ThB+DrwBo7O0kqBi311bo9l==", "total_rows": "15"}

***Note** enc_data: encrypted data will be decrypted using KEY and IV. See below for Data description step

**Sample Response Failure ( json )**

{"status":"2","api_txn": "NIC-HRY001-20250514-102008-876625",”message”:”No Record Found”}

{"status":"3","api_txn": "NIC-HRY001-20250514-102008-876625",”message”:”Invalid username”}

{"status":"4","api_txn": "NIC-HRY001-20250514-102008-876625",”message”:”Data available only for previous date.”}

***Note – enc_data Decryption step**

**1.** Take the enc_**key** and decode(Base64Decode) then Decrypt using **private key**. After decryption you will get KEY/IV concatenate with '#'.
**2.** Decode (Base64Decode) the **enc_data** then Decrypt using KEY/IV. **Key/IV** you have already get from above step(1).

*Data is encrypted using AES-256 encryption using Key and IV.*