पत्र संख्या-स्था0-6-सामान्य विविध पत्रावल-94(भाग-3)/I/555302/2026/राज्य कर

कार्यालय आयुक्त, राज्य कर, उत्तर प्रदेश

(स्थापना अराजपत्रित अनुभाग)

लखनऊ:: दिनांक:: 20-04-2026

समस्त जोनल अपर आयुक्त,

राज्य कर, उत्तर प्रदेश(जोन-गौतमबुद्ध नगर, गाजियाबाद-द्वितीय,

कानपुर-प्रथम को छोड़कर)।

विषयः-फैमिली आई0डी0 डाटाबेस से विभिन्न लाभार्थीपरक योजनाओं/सेवाओं के इंटीग्रेशन हेतु आधार हैश वैल्यू के उपयोग किये जाने के संबंध में ।

कृपया उपर्युक्त विषयक मुख्य सचिव, नियोजन विभाग, उत्तर प्रदेश शासन के पत्रांक संख्या-636/42/7-363/107944/ज0नि0प्र0/2024 दिनांक 17.03.2026**(छायाप्रति संलग्न)** का सन्दर्भ ग्रहण करने का कष्ट करें, जिसके द्वारा फैमिली आई0डी0 डाटाबेस से विभिन्न लाभार्थीपरक योजनाओं/सेवाओं के इंटीग्रेशन हेतु आधार हैश वैल्यू के उपयोग किये जाने के संबंध में दिशा-निर्देश जारी किये गये हैं ।

अतः नियोजन विभाग, उत्तर प्रदेश शासन के पत्रांक संख्या-636/42/7-363/107944/ज0नि0प्र0/2024 दिनांक 17.03.2026 की प्रति संलग्न कर इस आशय से प्रेषित किया जा रहा है कि पत्र में दिये गये दिशा-निर्देशानुसार आवश्यक कार्यवाही कराने का कष्ट करें।

संलग्नकः-विभागीय वेबसाइड पर अपलोड ।

Digitally signed by
SUNIL KUMAR VERMA
Date: 20-04-2026
16:39:45

(सुनील कुमार वर्मा)

अपर आयुक्त(प्रशासन) राज्य कर,

उत्तर प्रदेश, लखनऊ ।

पृष्ठांकन पत्र संख्या व दिनांक उक्त ।

प्रतिलिपिः-निम्नलिखित को सूचनार्थ एवं आवश्यक कार्यवाही हेतु प्रेषित ।

1. - अपर आयुक्त, राज्य कर, जोन-गौतमबुद्ध नगर, गाजियाबाद-द्वितीय एवं कानपुर-प्रथम को उक्त पत्र दिनांक 17.03.2026 की प्रति संलग्न कर इस अनुरोध के साथ प्रेषित कि पत्र में दिये गये दिशा-निर्देशानुसार आवश्यक कार्यवाही कराने का कष्ट करें।
2. - अपर आयुक्त(लेखा) राज्य कर, उत्तर प्रदेश, लखनऊ ।
3. - अपर निदेशक, राज्य कर अधिकारी प्रशिक्षण एवं शोध संस्थान, गोमती नगर, लखनऊ ।
4. - संयुक्त आयुक्त(आई0टी0) राज्य कर, मुख्यालय को एक प्रति विभागीय वेबसाइट पर अपलोड हेतु ।
5. - आहरण वितरण अधिकारी, राज्य कर मुख्यालय, लखनऊ ।

संलग्नकः-विभागीय वेबसाइड पर अपलोड ।

Digitally signed by
Akhilesh Kumar Singh
Date: 23-04-2026
13:42:49

संयुक्त आयुक्त(स्था0अरा0) राज्य कर,

मुख्यालय, लखनऊ ।

प्रेषक,

प्रमुख सचिव
नियोजन विभाग,
उत्तर प्रदेश शासन।

सेवा में,

1. समस्त अपर मुख्य सचिव/प्रमुख सचिव/सचिव, उत्तर प्रदेश शासन।
2. राज्य सूचना एवं विज्ञान अधिकारी, एन0आई0सी0, उत्तर प्रदेश।
3. हेड, स्टेट ई-गवर्नेंस मिशन टीम, उत्तर प्रदेश।

पत्रांक: 636 /42/7-363/107944/ज0नि0प्र0/2024 दिनांक: 17 मार्च, 2026

विषय: **फैमिली आई0डी0 डाटाबेस से विभिन्न लाभार्थीपरक योजनाओं/सेवाओं के इंटीग्रेशन हेतु आधार हैश वैल्यू के उपयोग किये जाने के संबंध में।**

महोदय/महोदया,

कृपया अवगत ही है कि प्रदेश में संचालित लाभार्थीपरक योजनाओं के बेहतर प्रबन्धन, समयबद्ध लक्ष्यीकरण, पारदर्शी संचालन, योजनाओं में पात्र लाभार्थियों का शत-प्रतिशत आच्छादन एवं जनसामान्य हेतु सरकारी सुविधाओं का सरलीकरण करने के उद्देश्य से "फैमिली आई०डी०-एक परिवार एक पहचान योजना" संचालित की जा रही है। उल्लेखनीय है कि राज्य/केन्द्र सरकार की 95 से अधिक लाभार्थीपरक योजनाओं/सेवाओं के डेटाबेस को ए०पी०आई० के माध्यम से फैमिली आई०डी० डाटाबेस से इंटीग्रेट किया जा चुका है।

यह विदित है कि विभागों की विभिन्न योजनाओं/सेवाओं के लाभार्थियों का डेटा आधार संख्या, आधार हैश, या फैमिली मेम्बर आई0डी0 का यूनिक आइडेंटिफ़ायर के रूप में उपयोग कर ए0पी0आई0 के माध्यम से फैमिली आई0डी0 डाटाबेस के साथ साझा किया जाता है। यू0आई0डी0ए0आई0 द्वारा अधिसूचित Aadhaar (Authentication and Offline Verification) Regulations, 2021 के अनुपालन में विभागों द्वारा लाभार्थियों के आधार ऑथेंटिकेशन के समय उनका आधार विवरण संग्रहित नहीं किया जाता है। परिणामस्वरूप विभागों को आधार संख्या को यूनिक आइडेंटिफ़ायर के रूप में उपयोग करते हुए अपनी योजनाओं/सेवाओं का डाटा फैमिली आई0डी0 से साझा करने हेतु लाभार्थियों का आधार विवरण संबंधित Authentication User Agency (AUA) के माध्यम से आधार वॉल्ट से निर्धारित शुल्क का भुगतान कर एक्सट्रैक्ट कराना पड़ता है। इससे प्रक्रियात्मक विलम्ब होने के साथ-साथ संबंधित विभागों पर अतिरिक्त वित्तीय भार पड़ता है, जिसके कारण योजनाओं/सेवाओं के लाभार्थीपरक डाटा का फैमिली आई0डी0 डाटाबेस से इंटीग्रेशन अनावश्यक रूप से विलम्बित हो जाता है।

उक्त स्थिति में Aadhaar Regulations के अनुपालन तथा फैमिली आई0डी0 डाटाबेस से समयबद्ध इंटीग्रेशन सुनिश्चित करने हेतु योजनाओं/सेवाओं के लाभार्थियों के प्रत्येक सफल आधार ऑथेंटिकेशन के समय संबंधित विभाग आधार की SHA-256 हैश वैल्यू जेनरेट कर उसे संग्रहित कर सकते हैं। तत्पश्चात विभागों द्वारा आधार हैश को यूनिक आइडेंटिफ़ायर के रूप में उपयोग करते हुए लाभार्थीपरक डाटा फैमिली आई0डी0 डाटाबेस के साथ साझा किया जा सकता है। SHA-256 हैश जेनरेट करने हेतु सैंपल कोड संदर्भ के लिए संलग्न है।

अतः आपसे अनुरोध है कि कृपया उपरोक्तानुसार कार्यवाही करने हेतु सम्बंधित को निर्देशित करने का कष्ट करें।

संलग्नक: यथोपरि।

JC विधा 6 दा/मार्च

अपर निदेशक (प्र0)
07.4.2026

भवदीय,
Digitally signed by ALOK KUMAR
Date: 17-03-2026 14:18:20
(आलोक कुमार)
प्रमुख सचिव

Adoption of Aadhaar Hash (SHA-256) for Family ID Reverse Seeding

Adoption of Aadhaar Hash (SHA-256) for Family ID Reverse Seeding

Jitendra Kumar Singh < jitendra.s@semt.gov.in >
Tue, 10 Mar 2026 7:20:06 PM +0530

To "Secretary UP"<pseoplan@nic.in>

Cc "vsplanningdpt888"<vsplanningdpt888@gmail.com>,"Neha Jain"<sc.ceg-up@gov.in>,"Director Lucknow"<dirmppd@nic.in>,"shivam.jyoti" <shivam.jyoti@semt.gov.in>,"trisha"<trisha@samagragovernance.in>

Dear Sir,

I hope this message finds you well.

As you are aware, during the initial design and conceptualization of the Family ID integration model with various data sources, the Family ID reverse seeding APIs were designed and developed using Aadhaar, Aadhaar Hash, or Family Member ID as the unique identifier to map beneficiary data with the Family ID.

However, in light of the stringent guidelines issued by UIDAI, Government of India, after Aadhaar authentication the encrypted Aadhaar must be securely stored in the Aadhaar Data Vault (ADV), while its reference key should be stored in the database of the concerned line department.

In order to avoid the presence of plain Aadhaar values in any digital ecosystem, it is highly advisable that the SHA-256 hash value of Aadhaar should also be generated and stored by the concerned department for every successful Aadhaar authentication. This measure will help departments map their beneficiary data with the Family ID using the Aadhaar Hash as a unique identifier.

The SHA-256 (Secure Hash Algorithm 256-bit) is a cryptographic hash function that converts any input data into a fixed-size 256-bit (32-byte) string, represented as a unique 64-character hexadecimal value. The hashing process is irreversible, meaning the original data cannot be reconstructed from the hash value. A sample code snippet for generating the SHA-256 hash is also enclosed for reference.

In view of the above, I humbly request you to kindly inform all concerned line departments to adopt this approach for Family ID reverse seeding using the Aadhaar hash value.

I shall be grateful for your kind consideration and support.

Regards,
Jitendra Kumar Singh
Head. SeMT – Uttar Pradesh

About Hash 256

Introduction:

SHA-256 (Secure Hash Algorithm 256-bit) is a cryptographic hash function that generates a fixed-size 256-bit (32-byte) hash value from input data of arbitrary size. It's a one-way function, meaning it cannot be reversed to obtain the original input data from the hash. It produces a unique, deterministic hash value for each unique input.

The length of a SHA-256 hash value is 256 bit, or 64 hexadecimal characters when represented in text form.

For Ex:

Input String: 123456789102

Generated Hash:
F3F44D808D917E8F38F7F7415C4BE2254B6BB268D85DF1C9A819B2222FA2C6D0

Below mentioned function is for Hash256 generation. Which take the raw data (Aadhaar) as input and returned its hash code.

Hash 256 Generation Code (C#):

```
using System;
using System.Security.Cryptography;
using System.Text;


public string ComputeSha256Hash(string inputAadhaar)
        {
            // Create a SHA256
            using (SHA256 sha256Hash = SHA256.Create())
            {
                // ComputeHash - returns byte array
                byte[] bytes =
sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(inputAadhaar));

                // Convert byte array to a string
                StringBuilder builder = new StringBuilder();
                for (int i = 0; i < bytes.Length; i++)
                {
                    builder.Append(bytes[i].ToString("x2"));
                }
                return builder.ToString();
            }
        }
```

Standard Operating Procedure (SOP) for Consumption of Family ID GetBeneficiaryHash API

1. Objective

The objective of this SOP is to define a standardized and secure mechanism for retrieval of **SHA-256 hashed Aadhaar values** for beneficiaries across three operational scenarios, namely:
(i) New beneficiary registration **(API not needed)**
(ii) Existing beneficiaries where Family ID and Member ID are available and
(iii) Existing beneficiaries where Family ID and Member ID is not available.

This SOP ensures that departments adopt the appropriate approach in each case—either generating the hash at their end or retrieving it through the GetBeneficiaryHash API—thereby maintaining data privacy, avoiding storage of plain Aadhaar, and enabling seamless integration with the Family ID ecosystem.

2. Overview

The **GetBeneficiaryHash API** enables departments to securely obtain:

- SHA-256 Hash of Aadhaar based on Family ID and Member ID or Unique Beneficiary ID

All request and response payloads are encrypted using **AES-256 (CBC mode)** to ensure end-to-end data security.

3. Key Operational Scenarios

The **GetBeneficiaryHash API** shall be utilized under the following three scenarios:

3.1 New Beneficiary Registration (Onwards)

- Applicable for all newly registered beneficiaries
- Department shall:
  - Capture Aadhaar number during its authentication
  - Generate **SHA-256 hash at department level** (Annexure 1)
  - Store only the **hashed Aadhaar** in scheme database

Important:

- **GetBeneficiaryHash** API call is **NOT required** in this case

3.2 Existing Beneficiary (Family ID Available)

- Applicable where **Family ID and Member ID are available**
- Department shall:
  - Call **GetBeneficiaryHash API**
  - Request Input:
    - SchemeCode (Mandatory)
    - FamilyId (Mandatory)
    - FamilyMemberId (Mandatory)
    - UniqueBeneficiaryID (Mandatory)
  - Response Output:
    - UniqueBeneficiaryID
    - SHA-256 hash of Aadhaar

3.3 Existing Beneficiary (Family ID Not Available)

- Applicable where **Family ID mapping is not available**
- Department shall:
  - Call **GetBeneficiaryHash API**
  - Request Input:
    - SchemeCode (Mandatory)
    - FamilyId (Empty String)
    - FamilyMemberId (Empty String)
    - UniqueBeneficiaryID (Mandatory)
  - Response Output:
    - UniqueBeneficiaryID
    - SHA-256 hash of Aadhaar

4. API Details

Parameter	Value
Endpoint	https://familyid.up.gov.in/IntegrationV3/api/Family/GetBeneficiaryHash
Method	POST
Content-Type	application/json
Security	AES-256-CBC Encrypted Request & Response

5. Request & Response Format

5.1 Encrypted Request Format

```
{
 "Data": "[ENCRYPTED_BASE64_STRING]"
}
```

5.2 Encrypted Response Format

```
{
 "Response": "[ENCRYPTED_BASE64_STRING]"
}
```

6. Payload Structures

6.1 Case: Using Family ID & Member ID

```
{
 "SchemeCode": "EJ2D1",
 "FamilyId": "213249012901",
 "FamilyMemberId": "21324901290103",
 "UniqueBeneficiaryId": "UBP0128102"
}
```

6.2 Case: Using Unique Beneficiary ID

```
{
 "SchemeCode": "EJ2D1",
 "FamilyId": "",
 "FamilyMemberId": "",
 "UniqueBeneficiaryId": " UBP0128103"
}
```

7. Decrypted Response Structure

7.1 Success Response

```
{
 "Status": "Success",
 "UniqueBeneficiaryID": "BEN12345",
 "SHA256": "HASH_VALUE"
}
```

Note: Hash Value i.e. "f482cde3093d34ae3c9e35f82b08a91be74175e243dc486e4e90fb433ab17485"

7.2 Error Response

```
{
 "Status": "Error",
 "Message": "Invalid Scheme Code"
}
```

8. Encryption/Decryption and Calling Method

Configuration

- Algorithm: AES-256
- Mode: CBC
- Padding: PKCS7
- IV: Base64 String
- Key: To be shared separately

Encryption / Decryption Method (.NET)

```
public string EncryptData(string iPlainStr, string iCompleteEncodedKey)
{
    AesCryptoServiceProvider aesEncryption = new AesCryptoServiceProvider();
    aesEncryption.KeySize = 256;
    aesEncryption.BlockSize = 128;
    aesEncryption.Mode = CipherMode.CBC;
    aesEncryption.Padding = PaddingMode.PKCS7;
    aesEncryption.IV = Convert.FromBase64String("eUgJhWbRBT5qPaMDS0ehng==");
    aesEncryption.Key = Convert.FromBase64String(iCompleteEncodedKey);

    byte[] plainText = Encoding.UTF8.GetBytes(iPlainStr);
    ICryptoTransform crypto = aesEncryption.CreateEncryptor();
    byte[] cipherText = crypto.TransformFinalBlock(plainText, 0, plainText.Length);
    return Convert.ToBase64String(cipherText);
}

public string DecryptData(string iEncryptedText, string iCompleteEncodedKey)
{
    AesCryptoServiceProvider aesEncryption = new AesCryptoServiceProvider();
    aesEncryption.KeySize = 256;
    aesEncryption.BlockSize = 128;
    aesEncryption.Mode = CipherMode.CBC;
    aesEncryption.Padding = PaddingMode.PKCS7;
    aesEncryption.IV = Convert.FromBase64String("eUgJhWbRBT5qPaMDS0ehng==");
    aesEncryption.Key = Convert.FromBase64String(iCompleteEncodedKey);
    ICryptoTransform decrypto = aesEncryption.CreateDecryptor();
    byte[] encryptedBytes = Convert.FromBase64String(iEncryptedText);
    return Encoding.UTF8.GetString(
        decrypto.TransformFinalBlock(encryptedBytes, 0, encryptedBytes.Length)
    );
}
```

Calling Method:

```
    public void TestSHA_Api()
    {
        string payload = @"{
    ""SchemeCode"": ""CWLOR"",
    ""FamilyId"": ""2136900177O9"",
    ""FamilyMemberId"": ""21369001770904"",
    ""UniqueBeneficiaryId"": ""
}";

        string encryptedPayload = EncryptData(
            payload,
            "OGN2YjA4YWIyc2QzOTgwODg5ZHdzZGpkOTA5M3FuYzg="
        );

        string finalBody = @"{
    ""Data"": """ + encryptedPayload + @"""
}";

        var client = new
RestClient("https://familyid.up.gov.in/IntegrationV3/api/Family/GetBeneficiaryHash");

        var request = new RestRequest(Method.POST);
        request.AddHeader("Content-Type", "application/json");

        request.AddParameter("application/json", finalBody,
ParameterType.RequestBody);

        IRestResponse response = client.Execute(request);

        string encryptedResponse = response.Content;

        JObject responseObj = JObject.Parse(encryptedResponse);
        string encryptedResult = responseObj["Response"]?.ToString();

        string decryptedResponse = DecryptData(
            encryptedResult,
            "OGN2YjA4YWIyc2QzOTgwODg5ZHdzZGpkOTA5M3FuYzg="
        );

        Response.Write("<b>Encrypted Response:</b><br/>" + encryptedResponse +
"<br/><br/>");
        Response.Write("<b>Decrypted Response:</b><br/>" + decryptedResponse);
    }
```

9. Conclusion

This SOP provides a secure and standardized framework for Aadhaar hash retrieval. By clearly defining the three operational scenarios, it ensures correct API usage, prevents misuse, and eliminates ambiguity during implementation. Adherence to the prescribed encryption standards and workflows will ensure data privacy, regulatory compliance, and seamless integration with the Family ID ecosystem.

About Hash 256

Introduction:

SHA-256 (Secure Hash Algorithm 256-bit) is a cryptographic hash function that generates a fixed-size 256-bit (32-byte) hash value from input data of arbitrary size. It's a one-way function, meaning it cannot be reversed to obtain the original input data from the hash. It produces a unique, deterministic hash value for each unique input.

The length of a SHA-256 hash value is 256 bit, or 64 hexadecimal characters when represented in text form.

For Ex:

Input String: 123456789102

Generated Hash:
F3F44D808D917E8F38F7F7415C4BE2254B6BB268D85DF1C9A819B2222FA2C6D0

Below mentioned function is for Hash256 generation. Which take the raw data (Aadhaar) as input and returned its hash code.

Hash 256 Generation Code (C#):

```
using System;
using System.Security.Cryptography;
using System.Text;


public string ComputeSha256Hash(string inputAadhaar)
    {
      // Create a SHA256
      using (SHA256 sha256Hash = SHA256.Create())
      {
        // ComputeHash - returns byte array
        byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(inputAadhaar));

        // Convert byte array to a string
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < bytes.Length; i++)
        {
          builder.Append(bytes[i].ToString("x2"));
        }
        return builder.ToString();
      }
    }
```

Standard Operating Procedure for Consumption of "Family ID Not Found API"

1. Objective

As the concerned departments are already aware with the process of Family ID Reverse Seeding API, this SOP is intended to define the additional workflow to be followed in cases where "Res_Code = '0001' (Aadhaar not found in Family ID database)" is received. In such scenarios, the departments shall mandatorily utilize the **FamilyIdNotFound** API to submit beneficiary demographic details for further processing and mapping in the Family ID system.

To define the standard process for handling cases where Aadhaar is not found in Family ID database and initiating demographic-based seeding via alternate API.

2. API Endpoint

https://familyid.up.gov.in/IntegrationV3/api/Family/FamilyIdNotFound
Method: POST
Content-Type: application/json

3. Trigger Condition

This SOP applies when response from **Family ID Reverse Seeding API** i.e.:

https://familyid.up.gov.in/integrations/api/Family/GetFamilyIdandMemberDetailByAadhaar

or

https://familyid.up.gov.in/integrations/api/Family/GetFamilyIdandMemberDetailByAadhaarSha256

or

https://familyid.up.gov.in/integrations/api/Family/GetFamilyIdandMemberDetailByAadhaarSha512

Returns:

```
{
 "ResponseStatus": [
  {
   "Res_Code": "0001",
   "Res_Message": "aadhar not found in family db"
  }
 ]
}
```

4. Payload Fields

Name, DOB, Gender, Mobile, UniqueBeneficiaryID (Department Database), FatherName, DistrictCode, DistrictName, BlockCode, BlockName, GPCode, GPName, TownCode, TownName, WardName, WardCode, SchemeCode, FY

Example:

```
{
 "Name": "RAM KUMAR",
 "DOB": "1985-01-01",
 "Gender": "M",
 "Mobile": "98XXXXXXXX",
 "UniqueBeneficiaryID": "BEN12345",
 "FatherName": "SHYAM LAL",
 "DistrictCode": "123",
 "DistrictName": "Lucknow",
 "BlockCode": "456",
 "BlockName": "Chinhat",
 "GPCode": "789",
 "GPName": "ABC",
 "TownCode": "101",
 "TownName": "Lucknow",
 "WardName": "Ward 10",
 "WardCode": "10",
 "SchemeCode": "STG01",
 "FY": "2025-26"
}
```

5. Encryption Standard and Calling Method

Advances Encryption Standard (AES)-256 (CBC), PKCS7 Padding, Fixed IV, Shared Key

Encryption Key: Will be shared separately

Encryption Method:

```
public string EncryptData(string iPlainStr, string iCompleteEncodedKey)
{
  AesCryptoServiceProvider aesEncryption = new AesCryptoServiceProvider();
  aesEncryption.KeySize = 256;
  aesEncryption.BlockSize = 128;
  aesEncryption.Mode = CipherMode.CBC;
  aesEncryption.Padding = PaddingMode.PKCS7;
  aesEncryption.IV = Convert.FromBase64String("eUgJhWbRBT5qPaMDS0ehng==");
  aesEncryption.Key = Convert.FromBase64String(iCompleteEncodedKey);
  byte[] plainText = Encoding.UTF8.GetBytes(iPlainStr);
  ICryptoTransform crypto = aesEncryption.CreateEncryptor();
  byte[] cipherText = crypto.TransformFinalBlock(plainText, 0, plainText.Length);
  return Convert.ToBase64String(cipherText);
}
```

Calling Method:

```
public void TestFamilyIdNotFoundData()
{
    // ✅ Proper JSON payload (use double quotes)
    string payload = @"{
""Name"": ""Ravi Kumar"",
""DOB"": ""1990-05-15"",
""Gender"": ""Male"",
""Mobile"": ""9876543210"",
""UniqueBeneficiaryID"": ""UBI123456789"",
""FatherName"": ""Shyam Kumar"",

""DistrictCode"": ""101"",
""DistrictName"": ""Bijnor"",
""BlockCode"": ""202"",
""BlockName"": ""Najibabad"",
""GPCode"": ""303"",
""GPName"": ""Jalalabad"",
""TownCode"": ""404"",
""TownName"": ""Najibabad Town"",
""WardName"": ""Ward 12"",
""WardCode"": ""12"",

""SchemeCode"": ""SCM001"",
""FY"": ""2025-26""
}";

        // ✅ Encrypt
        string data = EncryptData(payload,
"OGN2YjA4YWIyc2QzOTgwODg5ZHdzZGpkOTA5M3FuYzg=");

        // ✅ Correct API URL (your new API)

        var client = new
RestClient("https://familyid.up.gov.in/IntegrationV3/api/Family/FamilyIdNotFound");

        var request = new RestRequest(Method.POST);
        request.AddHeader("Content-Type", "application/json");

        // ✅ Correct JSON body
        string finalBody = @"{ ""Data"": """ + data + @""" }";

        request.AddParameter("application/json", finalBody, ParameterType.RequestBody);

        IRestResponse response = client.Execute(request);

        string JsonResponse = response.Content;
        Response.Write(JsonResponse);
    }
```

6. Response Codes

0000 → Data Inserted Successfully
0001 → Data Not Inserted
9999 → Exception/Error

Response Example:


```
{
 "ResponseStatus": [
  {
   "Res_Code": "0000",
   "Res_Message": "Data Inserted Successfully"
  }
 ]
}
```


7. Conclusion

This SOP ensures a standardized and seamless mechanism for handling cases where Aadhaar is not found in the Family ID database during reverse seeding. By mandatorily utilizing the **FamilyIdNotFound API**, departments can ensure inclusion of such beneficiaries through demographic-based identification. This process will improve overall data coverage, reduce exclusion errors, and support accurate beneficiary mapping in the Family ID ecosystem. Departments are required to strictly adhere to the defined workflow and security protocols to maintain data integrity and compliance.s