F. no. HQ-13064/1/2024-AUTH-I HQ/C-15014

Unique Identification Authority of India

(Authentication and Verification Division)

3rd floor, UIDAI Head Office
Bangla Sahib Road, Gole Market
New Delhi – 110001
Dated 31 July 2025

Circular 9 of 2025

To:

All Requesting entities (AUAs, KUAs, Sub-AUAs and Sub-KUAs) and ASAs

Subject: Aadhaar Status Notification Framework Documentation

Reference is invited to UIDAI Circular 1 of 2025 dated 1.1.2025 subjected "Execution of Supplementary Agreement or Agreement to supplement AUA Agreement under sub-regulation (3A) of regulation 9 of the Aadhaar (Authentication and Offline Verification) Regulations, 2021".

2. For availing the Aadhaar status verification service, requesting entities are directed to use the UIDAI's provided API framework for the same. Technical specification document for Aadhaar status verification is attached in Annexure I.

3. This issues with approval of competent authority.

[illegible]

(Sanjeev Yadav)
Director
Tel.: 011-23478609
Email: dir2.auth-hq@uidai.net.in

Copy to:

1. All UIDAI Regional Offices
2. Technology Centre, Bangalore

Annexure I

Unique Identification Authority of India (UIDAI)

Aadhaar Status Notification Framework Documentation Version 1.0.0

July-2025

Framework Overview

- **Title**: UIDAI Aadhaar Status Notification Framework
- **Description**: This API handles the subscription, notification and interrogation exchange for Aadhaar status.
- **Version**: 1.0.0

Description

1.1 UIDAI status notification framework is backed by Para 9 (3A) of the Aadhaar (Authentication and Offline Verification) Regulation, 2021 issued by UIDAI. This framework enables the requesting entities who desirous of ensuring update of status regarding an Aadhaar number previously submitted *has been subsequently omitted or deactivated or reactivated, the Authority shall send a subsequent digitally signed appropriate response, along with related technical details* may enter into this supplementary agreement with UIDAI.

1.2 The framework would be operationalised through the implementation of the following API interactions between AUA/Sub-AUA and UIDAI.

- **Subscription**: AUA/Sub-AUA to subscribe to the notification service of UIDAI. AUA/subAUA will call the UIDAI subscription endpoint to register the schedule to get the notification. *Refer to Section* ***2.1*** *for details on the subscription.*
- **Polling**: AUA/subSub-AUA AUA to poll UIDAI and get the list status change records at the scheduled interval.
- **Verification**: This API endpoint would be hosted by UIDAI to provide a mechanism to interrogate the status code and obtain the reason for rejection. *Refer to Section* ***3.1*** *for details on the interrogation.*

1.3 **Billing**. UIDAI will record the transaction for billing purposes on successful notification push to AUA/Sub-AUA . Billing would be recorded for every successful push. Cost of the transaction and invoicing is beyond the scope of the document.

API Overview

2.1 The API requests and responses are structured with three primary components:

- **Signature**: A cryptographic signature generated using HMAC (Hash-based Message Authentication Code) to ensure message integrity and authenticity.
- **Header**: Contains metadata about the message, including the version of the protocol, message identifiers, timestamps, action types, and other key details necessary to process the request and respond accordingly.
- **Message**: The core content, which includes subscription details.

Subscription API Overview

3.1 **Summary:** This endpoint subscribes to an AUA or ASA to receive Aadhaar status notifications. The service will notify the registered endpoint whenever there is a status change related to Aadhaar.

HTTP Method: POST

```
{
   "signature": "HMAC of the {header}+{message}", "header": {
      "ver": "1.0.0",
      "msgId": "12345",
      "msgTs": "2024-12-29T10:00:00Z", "ac":
      "AUA123",
      "sa": "SubAUA001",
      "action": "subscribe",
      "isMessageEncrypted": false, "lk":
      "LICENSEKEY123"
   },
   "msg": {
      "notifyEndpoint": "https://example.com/notify", "startDate":
      "2024-12-30",
      "schedule": "0 0 * * *"
   }
}
```

Parameters:

- signature: Base 64 of HMAC signature of {header}+{message}.
- header: Contains metadata about the request.
  - ver: Version of the API (e.g., "1.0.0").
  - msgId: Unique identifier for the message.
  - msgTs: Timestamp when the message was created.
  - ac: AUA code assigned by UIDAI.
  - sa: Sub-AUA code assigned by UIDAI.
  - action: Action to be taken, set to "subscribe".
  - isMessageEncrypted: Boolean flag for encrypted message.
- msg: Contains the subscription details.
  - notifyEndpoint: The endpoint where notifications will be sent.
  - startDate: The date after which notifications should start.

- schedule: Cron expression for scheduling the notification.

Header Parameters:

- ● X-Request-ID
  - **Type**: string
  - **Format**: uuid(36-byte UUID)
  - **Required**: Yes
  - **Description**: A unique identifier for the request. The value must be a 36-character UUID
- **Content-Type**: application/json

Responses:

- **200 OK**: Request was successfully processed.
- **400 Bad Request**: There was an issue with the input (e.g., invalid format, missing fields).
- **500 Internal Server Error**: There was an unexpected issue while processing the request.

Example Response

```
{

   "signature": "HMAC of the {header}+{message}", "header": {
      "ver": "1.0.0",
      "msgId": "12345",
      "msgTs": "2024-12-29T10:05:00Z", "ac":
      "AUA123",
      "sa": "SubAUA001",
      "action": "subscribe",
      "isMessageEncrypted": false
   },

   "msg": {
      "status": "success",
      "message": "Subscription successful"
   }

}
```

400 Bad Request

```
{

   "status": "failure", "error": {
      "code": "STS-GEN-001",
      "message": "Generic error: Invalid input format."
   }
}
```

500 Internal Server Error

```
{
    "status": "failure", "error": {
        "code":  "STS-GEN-001",
        "message": "Generic error occurred while processing the request."
    }
}
```

4.1 Polling API

UIDAI will host an endpoint which will be polled by the AUA/ASAs to fetch status change notification.

HTTP Method: POST

https://<<IP Address>>/uidstatus/ver/poll

4.1 Request in Encrypted Form

```
{
    "signature": "HMAC of {header}+{message}", "header": {
        "ver": "1.0",
        "msgId": "string",
        "msgTs": "ISO-8601 Timestamp", "lk": "ASA
        license key",
        "ac": "AUA Code",
        "sa": "Sub-AUA Code",
        "action": "notify", "isMessageEncrypted":
        true | false
    },
    "msg": {
        "txnId": "Unique transaction ID", "header":
        {
            "alg":  "AES-256-GCM",
            "enc":  "RSA-OAEP",
            "requestSessionKey":  "...",
            "thumbprint":  "...",
            "iv": "..."
        },
        "data": "<Base64Url-encoded ciphertext>", "requestHMAC":
        "encrypted hash of "msg" block"
    }
}
```

4.2 Preparation of the datafield.

```
{
        "ac": "AUA license key", "sa": "Sub-
        AUA license key",
        "lastPolledDate": "ISO 8601 date"
}
```

Steps to prepare the cipher datafield as follows:-

- ASA/AUA is required to prepare the above JSON object.
- In the next step, the JSON object is required to be marshalled to a byte array.
- Post marshalling to a byte array, the hash of the byte array is to be calculated. This hash is to be used to populate the value portion of the requestHMAC key.
- Then the byte array is encrypted using a symmetric key algorithm. The key to be
  generated to be preserved to encrypt in the subsequent stage. **AES-256-GCM** algorithm is to be used for symmetric key encryption.
- Encrypted data is encoded as base64 string and used as value for the data field.
- The symmetric key is encrypted using the public certificate of the UIDAI and then encoded using Base64 to get the value portion of requestSessionKey.

4.3 Response in Encrypted Form

```
{
    "signature": "HMAC of {header}+{message}", "header": {
        "ver": "1.0",
        "msgId": "string",
        "msgTs": "ISO-8601 timestamp", "lk": "ASA
        license key",
        "ac": "AUA Code",
        "sa": "Sub-AUA Code",
        "action": "notify",
            "isMessageEncrypted": false
        },
        "msg": {
            "header": {
                "alg":  "AES-256-GCM",
                "enc":  "RSA-OAEP",
                "requestSessionKey":
    "VGhpcyBpcyBhIHNhbXBsZSBlbmNyeXB0ZWQgc2Vzc2lvbiBrZXk=",
                "thumbprint": "abcdef1234567890abcdef1234567890abcdef12", "iv":
                "MTIzNDU2Nzg5MGFiY2RlZg=="   // base64url-encoded IV
            },
            "txnId": "Unique transaction ID"

"recordPending": "Total number of records pending for sync up",

"data":
    "U2FtcGxlIGVuY3J5cHRlZCBkYXRhIGJsb2Igd2l0aCB1aWQgc3RhdHVzIHJlc3  BvbnNl",
            "requestHMAC":  "YWFhYmJiY2NjZGRkZWVlZmZmZ2dnZ2dn"
        }
    }
```

4.4 Decoding of the datafield

```
    [
            {
                "referenceId": "Unique Reference ID", "uidToken":
                "Tokenized UID", "timestamp": "Status change
                timestamp", "status": "actv | susp | inactv"
            }
    ]
```

Steps to decode the datafield as follows:-

- The Base64 encoded encrypted symmetric key is decoded into byte array.
- The symmetric key is to be extracted from the value portion of the requestSessionKey. AUA/ASA to use their private key to decrypt the

session key.

- Decode the data field, which is in base64 string into a byte array.
- The Byte array is decrypted using the **AES-256-GCM** algorithm and the symmetric key.
- The hash of the byte array is computed and then compared with the requestHMAC.
- Decrypted Byte array is then marshalled into a JSON object as represented above.

4.5 Attribute Definition

The following attributes are used in status notification request - response.

- signature: HMAC signature of {header}+{message}.
- header: Contains metadata about the notification.
  - ver: API version.
  - msgId: Message ID.
  - msgTs: Timestamp when the message was created.
  - ac: AUA code.
  - sa: Sub-AUA code.
  - action: Set to "notify".
  - isMessageEncrypted: Boolean flag indicating whether the message is encrypted.
- msg: Contains the notification data:
  - txnId: Transaction ID for the notification event.
  - recordPending : Number of records pending for notification.
  - data: List of status change UIDs in tokenized form.
  - requestHMAC : HMAC of the cleartext byte array
  - header:
    i. alg: Defines the symmetric encryption algorithm used to encrypt the message. For example, AES (Advanced Encryption Standard) can be used in modes such as CBC (Cipher Block Chaining) or GCM (Galois/Counter Mode). **UIDAI at present supports only AES-256-GCM.**
    ii. enc: Specifies the asymmetric encryption algorithm used to encrypt the session key (e.g., RSA). Asymmetric encryption allows the session key to be securely shared between the sender and the receiver. **UIDAI at present supports only RSA-OAEP.**
    iii. requestSessionKey: The actual session key used to encrypt the message. The session key is encrypted with an asymmetric encryption algorithm (like RSA) and then Base64Url-encoded to ensure it's safely transmitted.
    iv. thumbprint: The thumbprint of the public key that was used to

encrypt the session key. This thumbprint ensures that the correct public key is being used and allows the recipient to verify the key used for encryption.

v. iv: The Initialization Vector used in encryption. The IV ensures that even if the same plaintext is encrypted multiple times, it will produce different ciphertexts each time. The IV is typically a random bit string used in modes like AES-CBC.

5.1 UID Status Verification API

The UID Status Verification API allows Authentication User Agencies (AUAs) and Sub-AUAs (Sub-Authentication User Agencies) to check the status of a UID (Unique Identification Number) or a list of UIDs after receiving status modification intimations from UIDAI. This API is designed to facilitate secure, real-time access to the status of UIDs to ensure accurate and up-to-date identity information for services requiring Aadhaar-based authentication.

The API request and response are structured with three primary components:

- **Signature**: A cryptographic signature generated using HMAC (Hash-based Message Authentication Code) to ensure message integrity and authenticity.
- **Header**: Contains metadata about the message, including the version of the protocol, message identifiers, timestamps, action types, and other key details necessary to process the request and respond accordingly.
- **Message**: The core content, which includes encrypted data, encryption details (such as algorithm and keys), and a session key to securely communicate the UID status check.

This specification outlines the message format, the cryptographic methods used, and the endpoint details, ensuring secure, efficient, and standard-compliant communication between parties using the UID Status Verification API. The API supports both encrypted and non-encrypted message exchanges, with the flexibility to accommodate varying use cases for UID status verification.

5.1.1 Sample Request

```
{
    "signature": "HMAC of the {header}+{message}", "header": {
        "ver": "1.0.0",
        "msgId": "12345",
        "msgTs": "2024-12-29T10:00:00Z",
        "action": "search",
        "tid": "registered",
        "ac": "AUA123",
        "sa": "SubAUA001", "lk":
        "LICENSEKEY123",
        "totalCount": 100,
        "isMsgEncrypted": false
    },

    "msg": {
        "header": {
            "alg": "AES-256-GCM",
            "enc": "RSA-OAEP",
            "requestSessionKey": "encrypted_session_key",
            "thumbprint": "thumbprint_string",
            "iv": "initialization_vector"
        },

        "data": "Base64Url-encoded encrypted payload",
        "requestHMAC": "HMAC of the encrypted message"
    }

}
```

Data Block before encryption:

```
[

        {

            "type": "uid",
            "uidToken": "123456789012"
        },

        {

            "type": "uid",
            "uidToken": "987654321098"
        },

        {

            "type": "token", "uidToken":
            "abcdef123456"
        }
    ]
```

5.1.2 Sample Response

```
    {
       "signature": "HMAC of the {header}+{message}", "header": {
          "version":  "1.0.0",
          "msgId":  "12345",
          "msgTs":  "2024-12-29T10:05:00Z",
          "action":  "search",
          "ac": "AUA123",
          "sa": "SubAUA001",
          "totalCount": 100,
          "isMsgEncrypted": false
       },
       "msg": {
          "header": {
             "alg": "AES with GCM",
             "enc": "RSA",
             "requestSessionKey": "encrypted_session_key",
             "thumbprint": "thumbprint_string",
             "iv": "initialization_vector"
          },
          "data": "Base64Url-encoded encrypted payload",
          "requestHMAC": "HMAC of the encrypted response"
       }
    }
```

Data Block post decryption

```
    [
       {
          "uidToken": "abcdef123456",
          "uidStatus": "ACTIVE",
          "errorCode": "0", "errorMsg":
          "No error"
       },
       {
          "uidToken": "987654abcdef123456321098", "uidStatus":
          "INACTIVE",
          "errorCode": "0",
          "errorMsg": "No error"
       },
       {
          "token": "abcdef123456656",
          "uidStatus": "",
          "errorCode": "STV-VER-001",
          "errorMsg": "UID status processing failed"
       }
    ]
```

5.2 Attribute Definition

The following attributes are defined within the UID Status Verification API for both the request and response. These attributes describe the structure and purpose of each part of the message for clarity and standardization.

- signature: HMAC signature of {header}+{message}.
- header: Contains metadata about the notification.
  - version: The version of the messaging protocol being used (e.g., "0.1.0").
  - msgId: A unique identifier for the message, used for tracking and correlation.
  - msgTs: Timestamp when the message was created, represented in ISO 8601 format.
  - action: The action to be performed (e.g., "search" for UID status check).
  - ac: Unique code identifying the AUA (Authentication User Agency).
  - sa: Code identifying the Sub-AUA (if applicable).
  - lk: The license key for the AUA or Sub-AUA.
  - totalCount: Number of records in the batch being processed.
  - isMsgEncrypted: Boolean flag indicating whether the message body is encrypted.
- msg:
  - header: Metadata for the encrypted message, including encryption settings and keys.
    - alg: The encryption algorithm used (e.g., "AES with GCM").
    - enc: Encryption method or mode used.
    - requestSessionKey: The session key used for the encryption.
    - thumbprint: A fingerprint of the public key used for encryption.

    - iv: The initialization vector for the encryption.
- data: The encrypted and base64-encoded payload containing the UID status check request.
    - uidToken: The UID token whose status is being returned (e.g., "123456789012" or a token value).
    - uidStatus: The current status of the UID (e.g., "ACTIVE", "INACTIVE", etc.).
    - errorFlag: An optional flag indicating if there was an error in processing the UID status. This could be used to return error details.
- requestHMAC: The HMAC used to validate the integrity of the request message.

6. Analytics Events

On successful notification of status change information to the AUA, an event would be generated, which will serve dual purpose i.e. invoicing with AUA/subAUA and business monitoring of the process. Event would comprise of following attributes:

- Event Metadata
    - _eventId : UUID value to uniquely identify the event
    - _eventType : “AADHAAR_STATUS_NOFICATION_PUSH”
    - _eventTimestamp: Event Timestamp in ISO Format
    - _version : Version of the Event Structure
- Payload
    - requestId : x-request-id of the transaction
    - msgId: message ID of the UIDAI Event
    - msgTs : Timestamp
    - refId : ReferenceId of ANH
    - ac : AUA Code
    - sa : subAUA Code
    - responseStatus : Response Status from AUA/subAUA