UIDAI

Unique Identification Authority of India
Planning Commission, Govt. of India (GoI),
3rd Floor, Tower II,
Jeevan Bharati Building,
Connaught Circus,
New Delhi 110001

AADHAAR OTP REQUEST API

API SPECIFICATION - VERSION 1.6

APRIL 2015

Table of Contents

1. INTRODUCTION 3
1.1 AUTHENTICATION FACTORS AND ONE TIME PIN (OTP) 3
1.2 TARGET AUDIENCE AND PRE-REQUISITES 4
1.3 OBJECTIVE OF THIS DOCUMENT 4
2. OTP REQUEST API 5
2.1 OTP USAGE SCENARIO 5
2.2 OTP REQUEST PROCESS 5
2.3 API PROTOCOL 6
2.3.1 *Element Details* *6*
2.4 API REQUEST DATA FORMAT 7
2.4.1 *Element Details* *7*
2.5 API RESPONSE DATA FORMAT 10
2.5.1 *Element Details* *10*

1. Introduction

The Unique Identification Authority of India (UIDAI) has been created, with the mandate of providing a Unique Identity (Aadhaar) to all Indian residents. The UIDAI proposes to provide online authentication using demographic and biometric data.

Aadhaar "*authentication*" means the process wherein Aadhaar Number, along with other attributes, including biometrics, are submitted to the Central Identities Data Repository (CIDR) for its verification on the basis of information or data or documents available with it. UIDAI will provide an online service to support this process. Aadhaar authentication service only responds with a "yes/no" and no personal identity information is returned as part of the response.

1.1 Authentication Factors and One Time Pin (OTP)

Authentication focuses on matching a person's identity based on the reliability of a credential offered. Various agencies have different requirements for the degree of assurance required while authenticating beneficiaries/customers. When authenticating a resident, multiple factors may be used to strengthen the authenticity of the request.

In general, following are the 3 categories of factors:

1. ***What you have***: Something the user uniquely has (e.g., a card, security token, mobile phone, access to email account, etc.)
2. ***What you know***: Something the user knows that is not public (e.g., a password, PIN, secret question, etc.). Demographic details such as date of birth may also be classified in this category although they are generally considered weak factors.
3. ***Who you are***: Something the user individually is or does (e.g., fingerprint pattern, iris pattern, signature, handwriting, etc.).

Aadhaar authentication provides multi-factor authentication to AUAs based on the following factors:

1. *What you have* – Mobile phone, email account
2. *What you know* –PIN, used ONLY for internal UIDAI transactions
3. *Who you are* – Fingerprints and Iris

When user agencies (AUAs) adopt Aadhaar authentication, they can choose option 1 or 3 above or both. Resident authentication can be strengthened by using multi-factor authentication.

To obtain OTP, following two ways can be used:

- By the resident her/himself using resident portal, by sending an inbound SMS from registered phone, by calling IVR from registered phone, or by using UIDAI provided mobile app running on registered phone.

- By programmatically initiating the request from the AUA/sub-AUA application in which authentication is used. Whichever application needing to validate OTP can thus initiate OTP request on behalf of the resident via an API. This document covers this API details for "***Application Initiated***" OTP request.

Notice that OTP is always delivered on the resident's mobile/email and application is expected to capture that OTP from the resident and pass it within the authentication request to authenticate the Aadhaar holder.

1.2 Target Audience and Pre-Requisites

This is a technical document and is targeted at software professionals working in technology domain and interested in incorporating Aadhaar authentication into their applications.

Readers must be fully familiar with following authentication documents published on UIDAI website (http://uidai.gov.in/auth) before reading this document.

1. Aadhaar Authentication Framework - http://uidai.gov.in/images/authDoc/d2_authentication_framework_v1.pdf
2. Aadhaar Authentication Operating Model - http://uidai.gov.in/images/authDoc/d3_1_operating_model_v1.pdf
3. Aadhaar Authentication API Specifications - http://uidai.gov.in/images/FrontPageUpdates/aadhaar_authentication_api_1_6.pdf

1.3 Objective of this document

This document provides Aadhaar OTP Request API specification for applications to request OTP on behalf of the resident. It contains details including API data format, protocol, and security specifications.

2. OTP Request API

This chapter describes the AUA/Sub-AUA application initiated OTP request API in detail including the flow, communication protocol, and data formats.

2.1 OTP Usage Scenario

Aadhaar authentication supports biometrics and/or One Time Pin (OTP) based authentication. While biometrics provide one factor (who you are), OTP provide another factor (what you have). Applications that take advantage of Aadhaar Authentication can use OTP to provide single factor authentication or along with biometrics for achieving two factor authentication.

When using OTP, at a high level, there are two distinct activities:

1. Resident obtaining the OTP on her/his phone/email. As described earlier, resident can obtain this directly from UIDAI servers via mobile app, inbound SMS, etc., or application can request an OTP to be sent to resident via this API.
    a. Applications should ask resident to enter OTP if the resident already has an OTP.
    b. If the resident does not have one, applications can initiate on behalf of the resident via this API.
2. Application requests the resident to provide the OTP. This OTP value should then be used within the Authentication API request by adding "Pv" element into the PID block (see Authentication API 1.6 Specification for details).

It's important to understand that the above two steps are two disconnected API calls and OTP value generated from using the OTP API (step 1 above) is then used as part of Authentication API to actually validate OTP via authentication (step 2 above).

Since OTP usage assumes that the resident is aware of the transaction (to be able to provide the OTP value received on his/her mobile/email to application), only a maximum of one OTP is valid for an Aadhaar number at any point in time. Every time a new OTP is generated, previous OTP, if any, cease to be valid. OTP generated has an expiry period for security reasons and it is expected that resident uses the OTP within that time. If not used, a new OTP needs to be generated for next transaction.

2.2 OTP Request Process

Application initiated OTP request works as follows:

1. Application (an application on an assisted device or self-service kiosks, or applications on the Internet), wanting to use Aadhaar OTP as a factor within Aadhaar authentication, initiates the transaction flow.
2. Application captures Aadhaar number along with any other data as needed.

3. Application, through AUA server, invokes the OTP Request API by forming digitally signed API Input XML (format explained later in this document).
4. UIDAI server processes the input, validates it, generates OTP, and sends it to resident's registered mobile/email.
5. UIDAI server then responds to OTP Request API with a response XML with success or indicating any error (see response XML details in later sections).
6. Resident receives the OTP from Aadhaar server on his/her email and/or mobile.
7. Application then requests resident to provide the OTP value so that application can send that via Aadhaar authentication API for authenticating the resident.

After receiving OTP, subsequent usage of OTP within Aadhaar authentication is explained in Aadhaar Authentication API specification 1.6. Notice that OTP expires with a UIDAI stipulated time. OTP message (SMS/Email) sent to resident provides time at which OTP was generated and when it is going to expire.

Applications should not initiate OTP request without the resident having to explicitly request the service offered by the agency and also should make sure that resident is made aware of OTP usage so that no OTP is sent to resident without his/her knowledge.

2.3 API Protocol

Aadhaar OTP Request API is exposed as stateless service over HTTPS. Usage of open data format in XML and widely used protocol such as HTTP allows easy adoption and deployment of these services.

Following is the URL format for Aadhaar OTP service:

```
https://<host>/otp/<ver>/<ac>/<uid[0]>/<uid[1]>/<asalk>
```

API input data should be sent to this URL as XML document using Content-Type "application/xml" or "text/xml".

As a best practice, for all SSL communications the agencies should automatically validate the SSL certificate and ensure it is validated against the revocation list online.

2.3.1 Element Details

host – Aadhaar OTP server address. Actual production server address will be provided to ASAs. Note that production servers can only be accessed through secure links. For development and testing purposes, public URL "*auth.uidai.gov.in*" can be used. ASA server should ensure that actual URL is configurable for flexibility.

Next part of the URL "otp" indicates that this is a OTP Request API call instead of regular authentication API call. Ensure that this is provided.

ver – OTP API version (mandatory). If not provided, URL points to current version. UIDAI may host multiple versions for supporting gradual migration. Version of this API is "1.6".

ac – A unique code for the AUA which is assigned by UIDAI. This is an alpha-numeric string having maximum length 10. (A default value "public" is available for testing.)

uid[0] and **uid[1]** – First 2 digits of Aadhaar Number of the resident for whom the OTP is being requested for. This is used for internal load-balancing.

asalk – A valid ASA license key. ASAs must send one of their valid license keys at the end of the URL. It is important that license keys are maintained safely. When adding license key to the URL, ensure it is "URL encoded" to handle special characters.

For all valid responses, HTTP response code 200 is used. All application error codes are encapsulated in response XML element. In the case of connection and other server errors, standard HTTP error response codes are used (4xx codes such as 403, 404, etc.). HTTP automatic redirects also should be handled by ASA server.

2.4 API Request Data Format

Aadhaar OTP service uses XML as the data format for input and output. To avoid sending unnecessary data, do not pass any optional attribute or element unless its value is different from default value. Any bad data or extra data will be rejected.

Following is the XML data format for OTP API:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Otp uid="" tid="" ac="" sa="" ver="" txn="" lk="" type="">
  <Opts ch=""/>
  <Signature>Digital signature of AUA</Signature>
</Otp>
```

2.4.1 Element Details

Element: **Otp** (mandatory)

- Root element of the input XML for OTP service.

Attributes:

- **uid** – (mandatory) – If requesting for an OTP against an Aadhaar number for authentication, provide 12 digit Aadhaar number in this field. But, if this API is used for obtaining a verification code for a mobile (e.g., for mobile update API), provide 10 digit mobile number (excluding country code, it is assumed to be

India for now). If mobile number is given instead of Aadhaar number in this field "type" attribute MUST BE set to "M".

- **tid** – (mandatory) For Registered devices, send its unique Terminal ID. For Public devices, value should be passed as "public".
- **ac** – (mandatory) A unique code for the AUA which is assigned by UIDAI during AUA registration process. This is an alpha-numeric string having maximum length 10. (A Default value "public" is available only for testing.)
- **sa** – (mandatory) A unique "Sub-AUA" code. AUAs are expected to manage these codes within their system and ensure uniqueness within their system. This allows auditing and business intelligence to be provided at SA level. If AUA and SA are same agency, use value of "ac" for this attribute. This is an alpha-numeric string having maximum length 10.
- **ver** – (mandatory) version of the OTP API. Currently only valid value is "1.6".
- **txn** – (mandatory) AUA specific transaction identifier. AUA can choose to pass this as part of input. This is returned as part of response as is. This is very useful for linking transactions full round trip across systems. This is an alpha-numeric string of maximum length 50. Only supported characters are A-Z, a-z, 0-9, period, comma, hyphen, backward & forward slash, left & right parenthesis, and colon. No other characters are supported. It is highly recommended that AUAs use this attribute for correlating requests with responses for auditing and verification.
- **lk** – (mandatory) A valid "License Key" assigned to the AUA. Administration portal of UIDAI will provide a mechanism for AUA administrator to generate license keys. This is an alpha-numeric string of length up to 64 characters.
- **type** – (optional) Type of OTP request. Possible values are:
    - "A" – uid attribute value shall refer to Aadhaar number (this is the default option). If this attribute is missing, it is assumed that "uid" field contains a valid Aadhaar number.
    - "M" – uid attribute value shall refer to new mobile number (this is used when a verification code needs to be sent to a mobile number, see mobile update API for details).

  Note: You can use any valid license key that has OTP feature enabled for this purpose.

Element: **Opts** (Optional)

- Various options are provided under this element.

Attributes:

- **ch** – (Optional) Valid only when using OTP against an Aadhaar number (when using Aadhaar number in "uid" attribute). Channel through which OTP should be sent. Possible values are:
    - "00" – send OTP via both SMS and Email (this is the default)
    - "01" – send OTP via SMS only
    - "02" – send OTP via Email only

  If type attribute is set to "M", notification shall be sent ONLY to the given mobile number via SMS. Any value in this "ch" attribute will be ignored.

Element: **Signature** (mandatory)

- The request XML should be digitally signed for message integrity and non-repudiation purposes.
- Digital signing should always be performed by the entity that creates the final request XML
    1. AUA can digitally sign after forming the API input XML. This is almost always the case. In such cases, AUA ensures the message security and integrity between AUA servers and its client applications.
    2. ASA can digitally sign the request XML if it is a domain-specific aggregator and forms the request XML on behalf of the AUA. In such cases, ASA and AUA ensure the message security and integrity between their servers.
- Procuring digital signature certificates:
    1. It should be procured from a valid certification authority as per Indian IT Act (see http://cca.gov.in/rw/pages/faqs.en.do#thecaslicensedbythecca)
    2. Digital certificates have two parts:
        - X.509 certificate representing public key.
        - Private Key which is used for digital signing. Private Key should be stored securely and is the responsibility of the owner of the certificate to ensure that it is not compromised.
    3. It should be a class II or class III certificate.
    4. X.509 certificate contains information about the owner of the certificate; in this case it will be details of the person and the organization to which he/she belongs. UIDAI server checks to ensure that certificate belongs to the ASA or AUA organization. Hence, it is mandatory that "O" attribute of "Subject" in the X.509 certificate matches the name of the organization.
- Digital signing of request XML
    1. XML digital signature algorithm as recommended by W3C.
    2. Signature should include key info element that contains X.509 certificate details. This is needed for UIDAI server to validate the signer.
- Verification of digital signature by UIDAI servers. UIDAI server validates the signature in the following sequence:
    1. Checks if the signature element is present. If not, it throws an error.
    2. If signature element is present, then it validates if the certificate is issued by one of the valid certification authority. If not valid, throws error.
    3. If it is a valid certificate, then it validates whether the "O" attribute in the X.509 certificate's subject matches the AUA organization name. If yes, proceeds with API logic.
    4. If it does not match AUA organization name, it checks configuration to see if ASA is allowed to sign on behalf of that AUA. If not, throws error.
    5. If ASA is allowed to sign on behalf of that AUA, it checks whether the "O" element of the certificate matches with the organization name of the ASA. If not, throws error.
    6. If it matches, it proceeds with API logic.
    7. In either case, once it is determined that request XML has been signed by either the AUA or ASA and is issued by a valid CA, UIDAI server will then check whether the certificate is white listed by AUA or ASA in the UIDAI Web portal.

- UIDAI web portal will optionally allow ASA and AUA users to upload public certificates used for signing.
- This provides for an option wherein ASA or AUA can choose to delete a certificate from the UIDAI portal in case it gets expired or is compromised. This acts as an additional security check.

2.5 API Response Data Format

Response XML is as follows:

```
<OtpRes ret="" code="" txn="" err="" ts="" info=""/>
```

2.5.1 Element Details

Element: **OtpRes**

Attributes:

- **ret** – Result of OTP generation request. "y" if successful, "n" if failure in which case "err" attribute will provide the reason of failure.
- **code** – Unique alphanumeric "OTP response" code having maximum length 40.
- **txn** – AUA specific transaction identifier. This is exactly the same value that is sent within the request.
- **ts** – Timestamp when the response is generated. This is of type XSD dateTime.
- **err** – Failure error code. If OTP request fails, this attribute provides any of the following codes (for updates, see https://developer.uidai.gov.in/site/api_err):
  - "**110**" – Aadhaar number does not have verified mobile/email
  - "**111**" – Aadhaar number does not have verified mobile
  - "**112**" – Aadhaar number does not have both email and mobile.
  - **"510"** – Invalid "Otp" XML format
  - **"520"** – Invalid device
  - **"521"** – Invalid mobile number
  - **"522"** – Invalid "type" attribute
  - **"530"** – Invalid AUA code
  - **"540"** – Invalid OTP XML version
  - **"542"** – AUA not authorized for ASA. This error will be returned if AUA and ASA do not have linking in the portal
  - **"543"** – Sub-AUA not associated with "AUA". This error will be returned if Sub-AUA specified in "sa" attribute is not added as "Sub-AUA" in portal
  - **"565" –** AUA License key has expired or is invalid
  - **"566" –** ASA license key has expired or is invalid
  - **"569"** – Digital signature verification failed
  - **"570"** – Invalid key info in digital signature (this means that certificate used for signing the OTP request is not valid – it is either expired, or does not belong to the AUA or is not created by a CA)
  - **"940"** – Unauthorized ASA channel
  - **"941"** – Unspecified ASA channel
  - **"950"** – Could not generate and/or send OTP
  - **"999"** – Unknown error

- **info** – Info block having masked mobile number and/or email to help application show appropriate message to resident. Length of "info" can be up to 256 characters and is composed of the following parts. Format of the info block is:
  <Version>{SHA-256 of Aadhaar Number, timestamp, OTP_api_ver, SHA-256 of ASA code, SHA-256 of AUA code, Sub-AUA code, masked-mobile, masked-email}
  - "Version" – is the version of the info structure. Currently "01"
  - "SHA-256 of Aadhaar Number" – is the SHA-256 hash value of Aadhaar Number provided as part of input
  - "SHA-256 of ASA code" – is the SHA-256 hash value of AUA code provided as part of input.
  - "SHA-256 of AUA code" – is the SHA-256 hash value of ASA code provided as part of input.
  - "Sub-AUA code" – same as the sub-AUA code passed in request.
  - "OTP_api_ver" – version of OTP Request API
  - "timestamp" – timestamp of the request.
  - "masked-mobile" – Masked mobile number. If this is not empty, AUA/Sub-AUA application can use this to display message "*OTP has been sent to mobile number <masked-mobile>"*
  - "masked-email" – Masked email id. If this is not empty, AUA/Sub-AUA application can use this to display message "*OTP has been sent to email ID <masked-email>"*