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 AUTHENTICATION

API SPECIFICATION - VERSION 1.6

JANUARY 2014

Table of Contents

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 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.

Before reading this document, readers are highly encouraged to read the following documents to understand the overall system:

1. UIDAI Strategy Overview - http://uidai.gov.in/UID_PDF/Front_Page_Articles/Documents/Strategy_Overview-001.pdf
2. The Demographic Data Standards and verification procedure Committee Report - http://uidai.gov.in/UID_PDF/Committees/UID_DDSVP_Committee_Report_v1.0.pdf
3. The Biometrics Standards Committee Report - http://uidai.gov.in/UID_PDF/Committees/Biometrics_Standards_Committee_report.pdf
4. From Exclusion to Inclusion with Micropayments - http://uidai.gov.in/images/FrontPageUpdates/microatmstandardsv1.3.pdf
5. Aadhaar-Communicating to a billion : An Awareness and Communication Report - http://uidai.gov.in/UID_PDF/Front_Page_Articles/Events/AADHAAR_PDF.pdf

Readers must also read the following related documents for complete understanding.

1. Aadhaar Best Finger Detection API - http://uidai.gov.in/images/FrontPageUpdates/aadhaar_bfd_api_1_6.pdf
2. Aadhaar OTP Request API - http://uidai.gov.in/images/FrontPageUpdates/aadhaar_otp_request_api_1_5.pdf
3. Aadhaar Registered Devices Specification - http://uidai.gov.in/images/aadhaar_registered_devices_1_0.pdf

1.2 Terminology

Authentication User Agency (AUA): An organization or an entity using Aadhaar authentication as part of its applications to provide services to residents. Examples include Government Departments, Banks, and other public or private organizations. All AUAs (Authentication User Agencies) must be registered within Aadhaar authentication server to perform secure authentication.

Sub-AUA (SA): An organization or a department or an entity having a business relationship with AUA offering specific services in a particular domain. All authentication requests emerging from an AUA contains the information on the specific SA. For example, a specific bank providing Aadhaar enabled payment transaction through NPCI as the AUA becomes the SA. Similarly, a state government being an AUA can have the health department under them as the SA using Aadhaar authentication while providing healthcare benefits.

Authentication Service Agency (ASA): An organization or an entity providing secure leased line connectivity to UIDAI's data centres for transmitting authentication requests from various AUAs. All connections to production authentication servers must come through private and secure connection through ASAs. Those AUAs who wish to provide their connectivity can become their own ASA where as smaller AUAs who do not wish to create direct leased line connection to UIDAI's data centres can use an ASA.

Terminal/Host Devices: Terminal/Host devices are devices employed by SAs/AUAs (both government and non-government) to provide services to the residents. Examples include MicroATM devices, PoS devices, PDS terminals, Smart phones, laptops, tablets, access control systems, etc. These devices will host the applications of the SA/AUA and support biometric capture mechanism to capture biometrics of residents for authentication purposes. Any additional features of these terminal devices would depend on specific needs of services offered by SAs/AUAs. Applications that uses Aadhaar Authentication on these devices must comply with specifications issues by UIDAI to protect all the biometric and demographic information provided by the residents.

Registered and Public Devices: Term "Registered Devices" refers to devices which are registered with Aadhaar system for encryption key management. Aadhaar authentication server can individually identify and validate these terminals and manage encryption keys on each registered device. Term "Public Devices" refers to devices which are not registered with Aadhaar system and uses its own encryption key generation scheme. Aadhaar authentication server does not individually identify public devices and uses an alternate encryption strategy for them.

Authentication Factors: Aadhaar authentication will support authentication using multiple factors. These factors include demographic data, biometric data, PIN, OTP, possession of mobile, or combinations thereof. Adding multiple factors may increase the

strength of authentication depending on the factors. Applications using Aadhaar authentication need to choose appropriate authentication factors based on the application needs. Currently, not all factors are supported.

Matching Strategy: Various demographic and biometric matchers use fuzzy matching and work on match thresholds and not on absolute digital (0 or 1) outputs, the interpretation of match scores to a MATCH or NON-MATCH needs to be tuneable using matching strategy. For demographic data matching, currently "Exact" and "Partial" matching strategies are supported in English and fuzzy matching of Indian language data is also supported.

1.3 Legal Framework

UIDAI has developed necessary legal framework and processes around Aadhaar authentication. These documents specify AUA/ASA registration process, security framework, operating model and covers their obligations, responsibilities and liabilities.

1.4 Objective of this document

This document provides Aadhaar Authentication API (Application Programming Interface) specification. It contains details including API data format, protocol, and security specifications.

For latest documents related to Aadhaar authentication, see http://uidai.gov.in/auth

2. Understanding Aadhaar Authentication

This chapter describes Aadhaar authentication, some of the envisioned usage scenarios, and working details. Technical details follow in subsequent chapters.

2.1 Aadhaar Number

The Unique Identification (Aadhaar) Number, which identifies a resident, will give individuals the means to clearly establish their identity to public and private agencies across the country. Three key characteristics of Aadhaar Number are:

1. Permanency (Aadhaar number remains same during lifetime of a resident)
2. Uniqueness (one resident has one ID and no two residents have same ID)
3. Global (same identifier can be used across applications and domains)

Aadhaar Number is provided during the initiation process called *enrolment* where a resident's demographic and biometric information are collected and uniqueness of the provided data is established through a process called *de-duplication*. Post de-duplication, an Aadhaar Number is issued and a letter is sent to resident informing the details.

2.2 Aadhaar Authentication at a Glance

Aadhaar authentication is the process wherein Aadhaar Number, along with other attributes, including biometrics, are submitted online to the CIDR for its verification on the basis of information or data or documents available with it.

In all forms of authentication the Aadhaar Number needs to be submitted so that authentication is reduced to a 1:1 match. In addition, Aadhaar authentication service only responds with a "yes/no" and no Personal Identity Information (PII) is returned as part of the response.

Aadhaar authentication provides several ways in which a resident can authenticate themselves using the system. At a high level, authentication can be using Demographics data and/or Biometric (FP/Iris) data, and/or OTP.

During the authentication transaction, the resident's record is first selected using the Aadhaar Number and then the demographic/biometric inputs are matched against the stored data within CIDR which was provided by the resident during enrolment/update process.

2.3 Aadhaar Authentication Usage

Aadhaar authentication enables agencies to verify identity of residents using an online and electronic means where the agency collects required information from the resident along with resident's Aadhaar Number and passes the same to UIDAI systems for verification.

Aadhaar authentication service provides services to instantly verify the identity of the resident against the available data in CIDR. Based on the needs of the service, different identifiers could be used along with Aadhaar Number. These identifiers could be combination of biometrics (such as fingerprints, iris impressions) and/or demographic information (such as Name, Date of birth, Address) and/or a secret PIN or OTP number known only to the resident.

Aadhaar Number along with certain demographic information such as name of the resident, date of birth, etc. helps to provides for simple authentication needs. For example, when MGNREGA beneficiary is enrolled and given a job card, resident could be biometrically authenticated against Aadhaar system to verify his/her Aadhaar number along with name, address.

Combination of Aadhaar Number and biometrics deliver online authentication without needing a card. During biometric authentication, agency will collect the Aadhaar Number along with one or more biometric impressions (e.g., one or more fingerprints, or iris impression alone or iris impression along with fingerprints). The information collected could then be passed to Aadhaar authentication server for authenticating the resident By combining biometric and demographic information stored with Aadhaar system, authentication of the resident could be strengthened. In addition, AUA specific factors such as cards may also be used in conjunction with Aadhaar authentication.

For further details on usage of Aadhaar in various service delivery scenarios, refer to "Aadhaar Enabled Service Delivery" White Paper published on UIDAI website - http://uidai.gov.in/images/authDoc/whitepaper_aadhaarenabledservice_delivery.pdf

2.4 Conclusion

Aadhaar authentication provides a convenient mechanism for all residents to establish their identity. It provides a national platform for identity authentication and can be used to deliver services effectively to residents across the country.

Rest of the document is technical in nature and is intended for software professionals working with applications wanting to enable their applications to support Aadhaar authentication.

3. Aadhaar Authentication API

This chapter describes the API in detail including the authentication flow, communication protocol, and data formats.

3.1 Authentication Flow

Following diagram explains various authentication scenarios and data flow.

Scenario 1 in the diagram is a typical authentication flow and is a case of an operator assisted transaction at a PoS terminal:

a) Resident provides Aadhaar Number, necessary demographic and biometric details to terminal devices belonging to the AUA/SA (or merchant/operator appointed by AUA/SA) to obtain a service offered by the AUA/SA.
b) Aadhaar authentication enabled application software that is installed on the device packages these input parameters, encrypts, and sends it to AUA server over either a mobile/broadband network using AUA specific protocol.
c) AUA server, after validation adds necessary headers (AUA specific wrapper XML with license key, transaction id, etc.), and passes the request through ASA server to UIDAI CIDR.
d) Aadhaar authentication server returns a "yes/no" based on the match of the input parameters.
e) Based on the response from the Aadhaar authentication server, AUA/SA conducts the transaction.

Scenario 2 below depicts the resident conducting assisted/self-service transactions with Aadhaar authentication on his/her mobile or on the Internet.

a) In this case, transaction data is captured on the mobile/Internet application provided by AUA/SA to residents
b) Resident provides necessary demographic data long with OTP (fingerprint/iris is also possible although not yet common on mobiles or PCs) in addition to AUA specific attributes (account number, password, PIN, etc.)
c) Step c, d, and e are same as in scenario 1 above.

Scenario 3 is a slight variant of 2nd scenario where AUA also plays the role of ASA and has direct connectivity to UIDAI data centers. Scenario 4 is how AUAs and application developers can test Aadhaar authentication using the public URL.

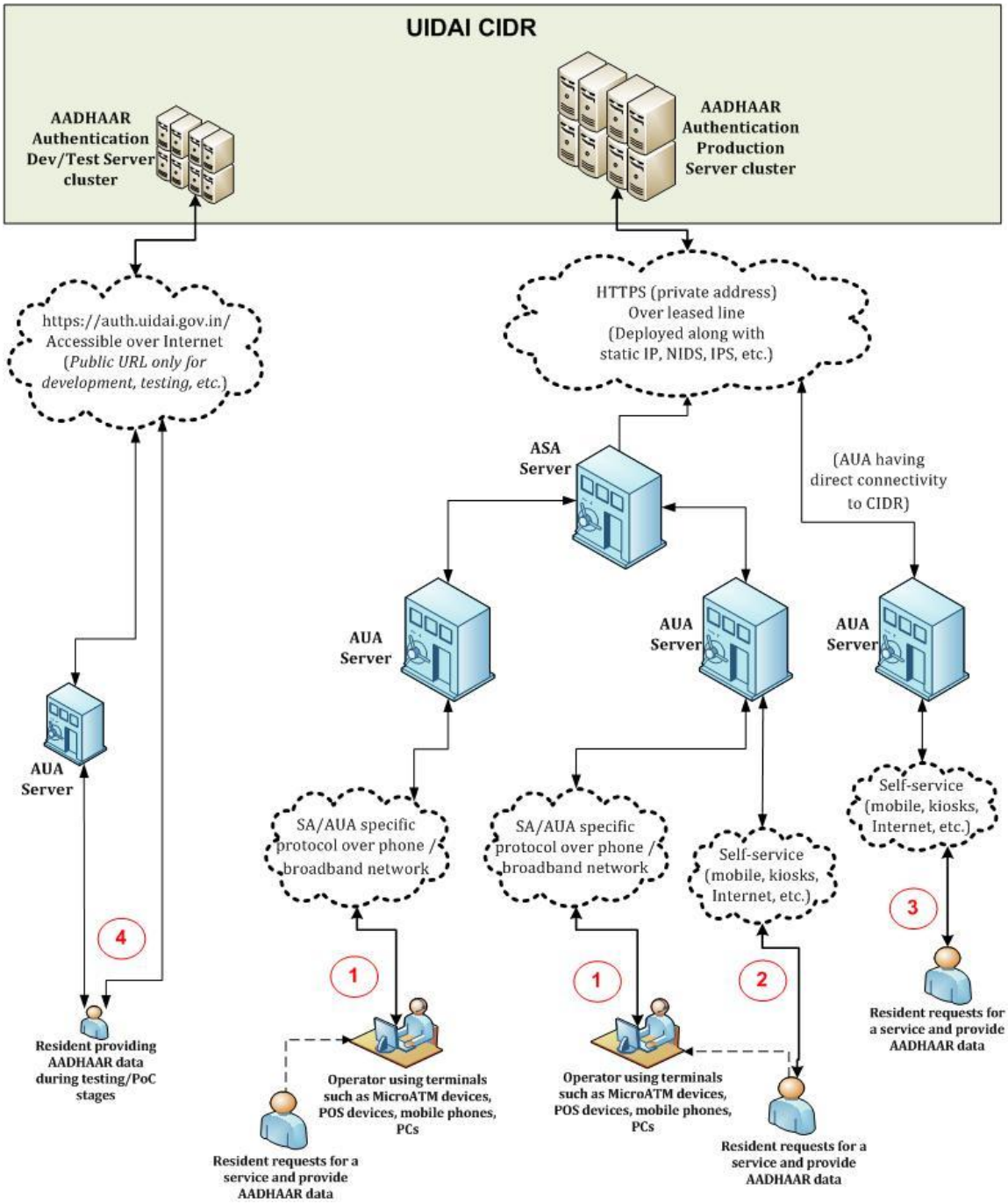

Figure 1: Aadhaar authentication flow under various scenarios

3.2 API Protocol

Aadhaar authentication service 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 Aadhaar authentication. To support strong end to end security and avoid request tampering and man-in-the-middle attacks, it is essential that encryption of data happens at the time of capture on the capture device.

Following is the URL format for Aadhaar authentication service:

```
https://<host>/<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".

For security reason data collected for Aadhaar authentication must not be stored in the devices or log files. It's essential for ASA and AUA to maintain audit records for all the authentication request metadata along with the response.

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.

3.2.1 Element Details

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

ver – Authentication API version (optional). If not provided, URL points to current version. UIDAI may host multiple versions for supporting gradual migration. As of this specification, default production version 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. Used for 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.

ASA server must send one of their valid license keys as part of the URL (see details above). Authentication related APIs are enabled only for valid ASAs and only for their registered static IP addresses coming through a secure private network.

3.3 Authentication API: Input Data Format

Aadhaar authentication 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 authentication API:

```
<Auth uid="" tid="" ac="" sa="" ver="" txn="" lk="">
  <Uses pi="" pa="" pfa="" bio="" bt="" pin="" otp=""/>
  <Tkn type="" value=""/>
  <Meta udc="" fdc="" idc="" pip="" lot="G|P" lov=""/>
  <Skey ci="" ki="">encrypted and encoded session key</Skey>
  <Data type="X|P">encrypted PID block</Data>
  <Hmac>SHA-256 Hash of Pid block, encrypted and then encoded</Hmac>
  <Signature>Digital signature of AUA</Signature>
</Auth>
```

"Data" element contains "Pid" (Personal Identity Data) element which is a base-64 encoded encrypted block. Complete "Data" block should be encrypted at the time of capture on the capture device. But, encoding (base-64) of "Data" block and packaging it with enveloping XML under "Auth" element can either be done on the device or on the AUA server based on the AUA needs. Device capability, protocol between devices and AUA server, and data format used between devices and AUA server, etc. should be considered for making that choice.

When using PID block in XML format (which is the default), following is the format for "Pid" element:

```
<Pid ts="" ver="">
  <Demo lang="">
    <Pi ms="E|P" mv="" name="" lname="" lmv="" gender="M|F|T" dob=""
dobt="V|D|A" age="" phone="" email=""/>
    <Pa ms="E" co="" house="" street="" lm="" loc=""
        vtc="" subdist="" dist="" state="" pc="" po=""/>
    <Pfa ms="E|P" mv="" av="" lav="" lmv=""/>
  </Demo>
  <Bios>
    <Bio type="FMR|FIR|IIR" posh="">encoded biometric</Bio>
  </Bios>
  <Pv otp="" pin=""/>
</Pid>
```

Instead of XML format, this version also allows PID block to be in binary format based on Protocol Buffers standard ((http://code.google.com/p/protobuf/). Notice that "Auth" XML must be in XML format. Binary format is only supported for PID block to enable smaller packet sizes to be transmitted from devices. See Appendix for details.

3.3.1 Element Details

Element: **Auth** (mandatory)

- Root element of the input XML for authentication service.

Attributes:

- **uid** – (mandatory) Aadhaar Number of the resident
- **tid** – (mandatory) For Registered devices, send its unique Terminal ID using "getTID()" function of the registered device (see Registered Devices section later in this document). 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 which is 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 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.
    - This **MUST NOT** start with "U*:" where "*" can be one or more alpha-numeric characters.
- **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 and use it within the authentication.
    - These license keys have expiry built into them and AUA administrator need to ensure that they generate new license keys before current ones expires through self-service portal.
    - This is an alpha-numeric string of length up to 64 characters.

Element: **Uses** (mandatory)

- This element specifies the authentication factors used by the request. When an authentication factor is specified in this element, that specific attribute must be present in the encrypted data block. This is particularly useful in situations where the AUA does not fully control the terminal device, but wishes to maintain a certain level of control on the authentication transaction.

Attributes:

- **pi** – (mandatory) Valid values are "y" or "n". If the value is "y" then at least one attribute of element "Pi" (part of "Demo" element) should be used in authentication. If value is "n", "Pi" element is not mandated.
- **pa** – (mandatory) Valid values are "y" or "n". If the value is "y" then at least one attribute of element "Pa" (part of "Demo" element) should be used in authentication. If value is "n", "Pa" element is not mandated.
- **pfa** – (mandatory) Valid values are "y" or "n". If the value is "y" then element "Pfa" (part of "Demo" element) should be used in authentication. If value is "n", "Pfa" element is not mandated.
- **bio** – (mandatory) Valid values are "y" or "n". If the value is "y" then at least one biometric element "Bio" (part of "Bios" element) should be used in authentication. If value is "n", "Bio" element is not mandated.
- **bt** – (mandatory only if "bio" attribute has value "y") provide a comma separated list of biometrics used. Valid values that can be used in this comma separated list are "FMR", "FIR", and "IIR". If "FMR" is part of the list, then at least one "Bio" element with type FMR should be used. Similarly, if "FIR" or "IIR" are part of the list, then at least one "Bio" element with those types must be used.
- **pin** – (mandatory) Valid values are "y" or "n". If the value is "y" then PIN should be used in authentication. Otherwise, "pin" is not mandated.
- **otp** – (mandatory) Valid values are "y" or "n". If the value is "y" then OTP should be used in authentication. Otherwise, "otp" is not mandated.

Element: **Tkn** (optional)

- This element can be used to provide a valid token details which is provided to resident such as mobile phone, NFC token, Smart Card, etc.. Currently, only the mobile number is considered a valid token.
- This is useful for adding second factor ("what resident has") for a self-service transaction from the resident's registered mobile phone. **This is for advanced usage only**.

  NOTE: If "Token" feature needs to be used, AUA must tie up with Telcos and derive the phone number of the phone from which the transaction originates.

Attributes:

- **type** – (mandatory) Type of the token. Currently only valid value is "001" representing mobile phone. UIDAI may add support for additional tokens such as NFC card in future.
- **value** – (mandatory) Token Number. Currently it must be the registered mobile number of the resident (without any prefix such as "0" or "+91") and must be derived from Telco network.

Element: **Meta** (Mandatory)

- This element specifies metadata related to the device and transaction. This is mandatory for better tracking, reporting, and trouble shooting.

Attributes:

- **udc** – (mandatory) Unique Host/Terminal Device Code. This is a unique code for the host device assigned within the AUA domain. This is an alpha-numeric string of maximum length 20.
    - This allows better reporting and tracking of devices as well as help resolve issues at the device level.
    - It is highly recommended that AUAs define a unique codification scheme for all their devices.
    - Suggested format is *"[vendorcode][date of deployment][serial number]*"
- **fdc** – (mandatory) Fingerprint device code. This is a unique code provided for the fingerprint sensor-extractor combination. AUAs should obtain this code from sensor providers for certified sensors. This is an alpha-numeric string of maximum length 10.
    - While using fingerprint authentication, this code is mandatory and should be provided. If the code is unknown or device is not certified yet, use "NC".
    - For non fingerprint authentication (when not using "bio" type "FMR" or "FIR"), use the value "NA"
- **idc** – (mandatory) Iris device code. This is a unique code provided for the iris sensor-extractor combination. AUAs should obtain this code from sensor providers for certified sensors. This is an alpha-numeric string of maximum length 10.
    - While using iris authentication, this code is mandatory and should be provided. If the code is unknown or device is not certified yet, use "NC".
    - For non iris authentication, use the value "NA"
- **pip** – (mandatory) Public IP address of the device. If the device is connected to Internet and has a public IP, then this must be populated with that IP address. If the device has a private IP and is behind a router/proxy/etc, then public IP address of the router/proxy/etc should be set. If no public IP is available, leave it as "NA".
- **lot** – (mandatory) Location type. Valid values are "G" and "P".
    - G – stands for geo coding in lat, long format
    - P – stands for postal pin code
- **lov** – (mandatory) Location Value.
    - If "lot" value is "G" then, this "lov" attribute must carry actual "lat,long,alt" of the location. Can be derived using GPS or using alternate method. Lat/Long should be in positive or negative decimal format (ISO 6709). Altitude is optional and may be populated if available. This can be of maximum length 39 (lat and long being maximum length of 15 in "*nnnnnnnnnn.nnnn*" format and alt being maximum length of 7 in "*nnnn.nn*" format with 2 commas in between them)
    - If "lot" value is "P" then, this "lov" attribute must have a valid 6-digit postal pin code.

Element: **Skey** (mandatory)

- Value of this element is base-64 encoded value of encrypted 256-bit AES session key. **Session key must be dynamically generated for every transaction**

(session key must not be reused) and must not be stored anywhere except in memory. See next chapter for encryption details.

Attributes:

- **ci** – (mandatory) Public key certificate identifier using which "skey" was encrypted. UIDAI may have multiple public keys in field at the same time. Value of this attribute is the certificate expiration date in the format "YYYYMMDD". Expiry date of the certificate can be obtained from the certificate itself.
- **ki** – (optional) **This is for advanced use only**. Do not use this attribute unless you clearly understand what you are doing. See chapter on "API and Data Security" for details.

Element: **Data** (mandatory)

- Contains the encrypted "Pid" element in base-64 format. See "Pid" element definition later.

Attributes:

- **type** – (optional) Type of the PID block format. It can have two values – "X" for XML and "P" for Protobuf binary format. Default value is assumed to be "X".

Element: **Hmac** (mandatory)

- Devices which is constructing the "Pid" element must perform the following to provide the Hmac value:
  - If Pid type is "X" (XML), then:
    - After forming Pid XML, compute SHA-256 hash of Pid XML string
    - Then encrypt using session key (skey)
    - Then encode using base-64 encoding (as described earlier, encoding can be done on the AUA server when forming final Auth XML)
  - If Pid type is "P" (Protobuf), then:
    - After forming Protobuf byte array for Pid, compute SHA-256 hash of Pid protobuf bytes.
    - Then encrypt using session key (skey)
    - Then encode using base-64 encoding (as described earlier, encoding can be done on the AUA server when forming final Auth XML)

Authentication server performs the following processing on each authentication request:

1. Decode and Decrypt the Pid from Data element. Based on type attribute of the "Data" element, the value of Data is either interpreted as XML or Protobuf bytes.
2. Re-compute the SHA-256 Hash of Pid.
3. Decode and decrypt the value of Hmac element.
4. Compare the re-computed SHA-256 hash with Hmac value received in authentication request.

a. If both values match, then, integrity of authentication request is preserved and server will proceed with further processing of the request.
b. If values do not match, reject the authentication request with error code representing "HMAC Validation failed".

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
  - 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.
  - 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:
  - 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)
  - 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.
  - It should be a class II or class III certificate.
  - 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
  - XML digital signature algorithm as recommended by W3C.
  - 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:
  - Checks if the signature element is present. If not, it throws an error.
  - 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.
  - 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.
  - 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.
  - 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.

- If it matches, it proceeds with API logic.
- In future, UIDAI may choose to conduct additional validations against white listed certificates within UIDAI database.

Element: **Pid** (mandatory)

Attributes:

- **ts** – (mandatory) Timestamp at the time of capture of authentication input. This is in format "YYYY-MM-DDThh:mm:ss" (derived from ISO 8601). Time zone should not be specified and is automatically defaulted to IST (UTC +5.30). **Since timestamp plays a critical role, it is highly recommended that devices are time synchronized with a time server**.

AUAs can buffer authentication requests and send it to Aadhaar authentication server to support occasional lack of network connectivity on the field. Maximum time up to which requests can be queued (buffered) will be defined by UIDAI policy. Currently, this will be configured to 24 hours. All requests with "ts" value older than this limit will be rejected.

When using SSK scheme or Registered Devices, this value is 4 hrs and all requests should be sent in same order as captured within 4 hours. Otherwise, requests will be rejected to ensure strong security.

- **ver** – (mandatory) version of the "Pid" element. Currently only valid value is "1.0". Notice that this is NOT same as Auth API version. Since authentication devices are responsible for generation of "Pid" XML, and there is a possibility that if there is a change in the structure of "Pid" XML, all the devices may not upgrade to latest software version.

Element: **Demo** (optional)

- Contains child elements "Pi", "Pa" and "Pfa", all of which are optional.
- All demographic data fields as per KYR specifications.

Attributes:

- **lang** – (optional) "Indian Language Code" in the case of using Indian language data for demographic match (see lname, lav attributes). This must be a valid language code from the following table

Language	Language code
Assamese	01
Bengali	02
Gujarati	05
Hindi	06
Kannada	07
Malayalam	11
Manipuri	12

Marathi	13
Oriya	15
Punjabi	16
Tamil	20
Telugu	21
Urdu	22

NOTE: Indian language matching of name and address allows data to be matched in any of the above languages using a fuzzy matching logic. In the case of address where multiple fields are provided as a single string (using "lav" attribute), it is recommended to separate each field (house, street, locality, vtc, district, etc) by comma.

Element: **Pi** (Optional)

- This element captures attributes related to "Personal Identity"

Attributes:

- **ms** – (optional) "Matching Strategy" for "name" attribute. Valid values are "E" (Exact match) and "P" (Partial match). This is used only for "name" attribute. Defaulted to "E".
- **mv** – (optional) "Match value" for "name" attribute. Valid values are the integers in the range 1 – 100 (inclusive). Default value is "100". "mv" attribute value MUST be specified when matching strategy ("ms" attribute) is "P".

  It represents the percentage of full words from the name stored in Aadhaar database that must be specified in the "name" attribute for the match to be considered successful. See examples below as part of "name" attribute description.

- **name** – (optional) Name of the resident in English. Maximum length is 60.

  NOTE: If "ms" and/or "mv" are provided, but, "name" attribute is not provided or empty value is provided, no name matching will be performed.

  When using matching strategy "Exact" (ms="E"), the name attribute is compared for exact match with the name stored in Aadhaar database. Though comparison is case insensitive, all the words of the name must be specified in the exact same order as provided by the resident during Aadhaar enrolment.

  When using matching strategy "Partial" (ms="P"), the name attribute is compared for partial match with the name stored in Aadhaar database based the following rules:

  1. Words from the name can appear in any order in the "name" attribute.
     For example: If resident's name is stored as "Anil Kumar Singh" in Aadhaar database, then, any of the inputs - "Kumar Anil Singh", "Anil Singh Kumar", "Singh Kumar Anil" or any other combinations – will result in successful match.

2. Usage of specific titles is allowed in the "name" attribute. These are ignored for matching purposes. Supported titles are "Mr", "Mrs", "Dr", and "Ms". No other titles are currently supported.
   For example: If resident's name is stored as "Anita Agarwal" in Aadhaar database, then, any of the inputs – "Dr. Anita Agarwal", "Ms. Anita Agarwal" or "Mrs Anita Agarwal" - will result in successful match,

3. Following special characters, if present in the "name" attribute, are ignored during matching:
   a. Period - (.)
   b. Comma (,)
   c. Hyphen (-)
   d. Asterisk (*)
   e. Opening and closing braces - '(' and ')'
   f. Opening and closing square brackets – '[' and ']'
   g. Apostrophes – `
   h. Single quotes – '
   i. Double quotes – "
   j. Forward slash - /
   k. Backward slash - \
   l. Hash - #
   m. Leading, trailing, and more than 1 contiguous spaces are removed before matching

   For example: If resident's name is stored as "Anita Agarwal" in Aadhaar database, then, "Agarwal, Anita" will result in successful match.

4. Input should not contain any additional words or initials that are not present in Aadhaar database. This will result in unsuccessful match and authentication failure.
   For example: If resident's name is stored as "Anita Agarwal" in Aadhaar database, then, "Anita Kumari Agarwal" or "Anita Kumari" or "Anita K Agarwal" will result in unsuccessful matches as the words "Kumari", "K" are not present in the Aadhaar database.

5. When used with "mv" value other than 100, then, inputs can have some words omitted or initials can be used in place a full word. The match is considered successful as long as input contains minimum number of matching full words as determined by the value of "mv".
   For example: Let us say that resident's name is stored as "Anil Kumar Singh" in Aadhaar database, and "mv" value is specified as 60 percent. It means at least 60% of total words must be matched. In the case of "Anil Kumar Singh", 60% of 3 (total words in name) is 1.8, rounded up to 2. Thus, any of the following inputs will result in successful match:
   a. "Anil Singh" – matches because of minimum 2 full words
   b. "Singh, Kumar Anil" – matches because of minimum 2 full words in any order and comma (,) is ignored.
   c. "Anil K Singh" – matches because of minimum 2 full words in any order and "K" is an initial of "Kumar".

Following input will result in unsuccessful match:

a. "Anil" – does not match because number of words is less than specified minimum.
b. "Anil K S" – does not match as initials are not counted as full words and hence, number of matching full words is less than specified minimum.
c. "Anil P Singh" – does not match as "P" is not an initial of any of the words stored in database.
d. "Anil S Singh" – does not match as after matching full words "Anil" and "Singh", "S" does not match as initial of "Kumar".

- **lname** – (optional) Resident's name in Indian language. This is a Unicode String in the language specified by the "lang" attribute of "Demo" element. Notice that this is a phonetic matching against the data stored in CIDR.

  NOTE: If this attribute is provided, "lang" attribute must be specified for "Demo" element.

- **lmv** – (optional) Local Language Match Value to adjust phonetic match threshold. This is a value between 1 and 100 (both inclusive).
- **gender** – (optional) Valid values are "M" for male, "F" for female, and "T" for transgender.
- **dob** – (optional) Date of Birth in "YYYY-MM-DD" format. If only year needs to be authenticated, then use format "YYYY".
- **dobt** – (optional) Date of Birth Type as indicated in Aadhaar system. This attribute can have only 3 values – "V" (for Verified), "D" (for Declared), and "A" (Approximate). When the resident is enrolled, DoB may be recorded along with any of these types.
- **age** – (optional) In certain use cases such as checking whether a resident can be considered a senior citizen or whether a resident is an adult (age above or equal to 18 years), it may be desired that only age of a resident can be verified using Aadhaar Authentication instead of verifying against complete data of birth. When "age" attribute is specified, authentication will pass if resident's age is "equal to or greater than" the input age. Else, it will fail with appropriate authentication error code.
- **phone** – (optional) Registered mobile phone number of the resident.
- **email** – (optional) Registered email address of the resident. This is case-insensitive match removing trailing and leading spaces.

Element: **Pa** (Optional)

- This element captures attributes related to "Personal Address". These are address fields as provided by the resident during enrolment or later updates. Only attributes that are sent as part of input will be compared.
- This element should not be used when using "Pfa" element as "Pa" and "Pfa" are mutually exclusive.

Attributes:

All attributes are compared case insensitive after leading and trailing spaces are trimmed and all the occurrences of consecutive spaces are replaced with single space.

- **ms** – (optional) "Matching Strategy" for address attributes. Only the value "E" (Exact match) is supported. This is used only when at least one address attribute is specified.
- **co** – (optional) "Care of" person's name.
- **house** – (optional) House identifier.
- **street** – (optional) Street name.
- **lm** – (optional) Landmark if any.
- **loc** – (optional) Locality where resident resides.
- **vtc** – (optional) Name of village or town or city.
- **subdist** – (optional) Sub-District name.
- **dist** – (optional) District name.
- **state** – (optional) State name.
- **pc** – (optional) Postal pin code.
- **po** – (optional) Post Office name.

Element: **Pfa** (Optional)

- This element captures attributes related to "Personal Full Address". It corresponds to the address of the resident as present in enrolment receipt or Aadhaar letter.
- This element should not be used when using "Pa" element as "Pa" and "Pfa" are mutually exclusive.

Attributes:

- **ms** – (optional) "Matching Strategy" for address attributes. Valid values are "E" (Exact match) and "P" (Partial match).
- **mv** – (optional) Valid values are integers in the range 1 -100 (inclusive). Default value is "100". It is used only when matching strategy (ms attribute) is "P" (Partial match).
  It represents the percentage of full words from the address stored in Aadhaar database that must be specified in the "av" attribute for the match to be considered successful,
- **av** – (optional) Resident's full address specified as a single string value.

  Normalization:
  "av" value and the resident's address stored in Aadhaar database, both are normalized using following rules before comparison.

  1. Following characters/phrases are ignored:
     a. Period - (.)
     b. Comma (,)
     c. Hyphen (-)
     d. Asterisk (*)

e. Opening and closing braces - '(' and ')'
f. Opening and closing square brackets – '[' and ']'
g. Apostrophes – `
h. Single quotes – '
i. Double quotes – "
j. Forward slash - /
k. Backward slash - \
l. Hash - #
m. Care of labels – C/O, S/O, D/O, W/O, H/O
n. Other labels - "No."

2. Leading and trailing spaces are trimmed and all the occurrences of multiple consecutive spaces are replaced with single space.

When using matching strategy "Exact" (ms="E"), the normalized "av" attribute is compared for exact match with the normalized resident's address stored in Aadhaar database.

When using matching strategy "Partial" (ms="P"), the normalized "av" attribute is compared for partial match with the normalized resident's address stored in Aadhaar database. Following are the rules of partial match:

1. Words can appear in any order.
2. Following additional normalizations are applied to both the "av" value and address stored in Aadhaar database:
   a. Commonly used words are replaced with their shortened version:
      I. "apartment" => "apt"
      II. "street" => "st"
      III. "road" => "rd"
      IV. "main" => "mn"
      V. "cross" => "crs"
      VI. "sector" => "sec"
      VII. "opposite" => "opp"
      VIII. "market" => "mkt"
   b. Suffixes typically used with numbers such as – st, nd, rd, th - are removed.
3. Example: 21st is converted to 21, 44th is converted to 44, etc. When used with "mv" value other than 100, some of the words can be omitted in the input. Match is considered successful if minimum number of full words that must match, as determined by the "mv" value, are present in the input.

Examples:
Scenario: Matching strategy "P", mv="60" (60%)

Given following value as resident's address stored in database,

c/o A K Singh, Apartment #12, Lake view colony, Main street, Near Swimming pool, Rajajinagar, Bangalore, Karnataka, 560055

Following will be the normalized address.

a k singh apt 12 lake view colony main st near swimming pool rajajinagar bangalore karnataka 560055

Following are examples of matching and their result:

1. "*s/o A K singh, Lake view colony, apt #12, main st, Karnataka - 560055*" will be normalized to "*a k singh lake view colony apt 12 main st karnataka 560055*" – which results in successful match as 12 words are matched (more than 11 which is rounded 60% of total words 17).
2. "*s/o A K singh, Lake view colony Bangalore 560055" will be normalized to "a k singh lake view colony Bangalore 560055*" – which results in unsuccessful match since only 8 words are matched where as a minimum of 11 was requested (60% match value).

- **lav** – (optional) Resident's Address in Indian language. This is similar to "av" attribute described above except that this is represented in an Indian language. This will be matched against address data available in the CIDR database. If this attribute is provided, "lang" attribute must be specified for "Demo" element.
- **lmv** – (optional) Local Language Match Value to adjust phonetic match threshold. This is a value between 1 and 100 (both inclusive).

Element: **Bios** – (optional)

- This element can have one or many "Bio" elements carrying biometric records to be matched.

Element: **Bio** (optional)

- If XML data format is used for PID block, this element contains single base 64 encoded biometric record. This is typically in plain ISO data format.
- In the case of registered devices, this contains encrypted FMR/FIR/IIR record (see "Using Registered Devices" section later in the document).

Attributes:

- **type** – (mandatory) This attribute specifies type of the biometric. Valid values are "FMR" (Finger Minutiae), "FIR" (Finger Image), and "IIR" (Iris Image).
  - ***FMR*** - The biometric data is of type "Fingerprint Minutiae Record". This data is in ISO minutiae format with no proprietary extensions allowed.
  - ***FIR*** - The biometric data is of type "Fingerprint Image Record". The data is a fingerprint image packaged in ISO 19794-4 format, which could contain a compressed or uncompressed image, of type PNG, WSQ, or Jpeg2000.
  - ***IIR*** - The biometric data is of type "Iris Image Record". The data is an iris image packaged in ISO 19794-6 format, which could contain a compressed (or uncompressed) image having type PNG or Jpeg2000.
- **posh** – (mandatory) In general, it is highly recommended that applications pass "UNKNOWN" unless it clearly knows which finger was used. Valid values are:

LEFT_IRIS
RIGHT_IRIS
LEFT_INDEX
LEFT_LITTLE
LEFT_MIDDLE
LEFT_RING

LEFT_THUMB
RIGHT_INDEX
RIGHT_LITTLE
RIGHT_MIDDLE
RIGHT_RING
RIGHT_THUMB
UNKNOWN

Element: **Pv** (optional)

- This element ("Pin Value") allows support for additional factors "pin" and "otp".

Attributes:

- **pin** – (optional) Actual value of PIN as set by resident. This attribute contains a 6 digit numeric value. **This option is NOT available for AUAs and is restricted to internal UIDAI usage only**.
- **otp** – (optional) Most recently generated One Time Pin (OTP) value for resident (Note that one resident can ONLY have one active OTP value). Unlike PIN, OTP is a one-time usage token and expires if not used within few minutes. See Aadhaar OTP API for details on requesting OTP.

As per UIDAI security policy, if number of failed attempts crosses upper limit, Aadhaar record may be put on hold. The upper limit is dynamically computed based on various heuristics and is not a static number.

3.4 Authentication API: Response Data Format

Authentication API does not provide any identity data as part of the response. All it does is to match given input and respond with a "yes/no". Response XML is as follows:

```
<AuthRes ret="y|n" code="" txn="" err="" ts="" actn="" info="">
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod
        Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha256" />
      <Reference URI="">
        <Transforms>
          <Transform
            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature" />
        </Transforms>
        <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>
        <DigestValue></DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue></SignatureValue>
  </Signature>
</AuthRes>
```

Agencies that use the authentication response need a mechanism to validate the authenticity of the authentication response for non-repudiation purposes. In order to enable verification and audit, authentication response will be digitally signed by UIDAI and the signature will be part of the response. AUAs are expected to preserve the entire response XML for non-repudiation purposes.

3.4.1 Element Details

Element: **AuthRes**

Attributes:

- **ret** – this is the main authentication response. It is either "y" or "n".
- **code** – unique alphanumeric "authentication response" code having maximum length 40. If the input is "not" processed due to errors such as decryption, wrong hmac value, etc., a value of "NA" will be returned. Once the integration is tested, this should not happen. But, due to some transmission errors or changes in deployments, if code is returned as "NA", AUAs may retry the transaction and if it continues to fail, may report to UIDAI technical support.
- **txn** – Authenticator 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.
- **actn** – (optional). This attribute may or may not exist in response. This attribute, alpha-numeric of max length 5, provides specific action codes (published from time to time) **meant to be shown to resident/operator**.
    - One possible use is to provide personalized resident feedback to improve authentication outcomes and required data update notifications.
    - **This attribute MUST be sent to front-end application by KSA and KUA to ensure action and corresponding message is displayed to resident/operator**.
- **info** – This attribute provides meta information on the details included in auth. This can be up to 128 characters and is composed of the following parts:

  Version 2.0 structure of "info" (latest):

  ```
  <Version>{SHA-256 of Aadhaar Number, SHA-256 of Demo element, Encoded Usage Data, pid_version, timestamp, fmrcount, fircount, iircount, auth_api_ver, SHA-256 of ASA code, SHA-256 of AUA code, SUB AUA code, lot, lov, lang, pi-ms, pi-mv, pi-lmv, pa-ms, pa-mv, pa-lmv, pfa-ms, pfa-mv, pfa-lmv, SHA-256 of tid}
  ```

  - "Version" – is the version of the info structure, "02"
  - "SHA-256 of Aadhaar Number" – is the SHA-256 hash value of Aadhaar Number provided as part of input
  - "SHA-256 of Demo Element" –
    - When Pid block is of type "X" (XML), then, it is the SHA-256 hash value of the substring representing Demo element in the Pid XML string.

  - When Pid block is of type "P" (Protobuf), then, it is the SHA-256 hash of the Protobuf value (byte array) of the Demo element.
- "Encoded Usage Data" - is 48 bit representation in HEX format of the various attribute usage of this authentication request.

  For version 2.0 of this block, Hexadecimal digits within the "Encoded Usage Data" should be interpreted based on below rules:

  1st hexadecimal digit:
  Bit 3-0: Version number of encoding. It will be hexadecimal "1" (binary: 0001) for encoding specified in this document.

  2nd hexadecimal digit:
  Bit 3: Was "`Pi->name`" attribute used?
  Bit 2: Was "`Pi->lname`" attribute used?
  Bit 1: Was "`Pi->gender`" attribute used?
  Bit 0: Was "`Pi->dob`" attribute used?

  3rd hexadecimal digit:
  Bit 3: Was "`Pi->phone`" attribute used?
  Bit 2: Was "`Pi->email`" attribute used?
  Bit 1: Was "`Pi->age`" attribute used?
  Bit 0: Was "`Pa->co`" attribute used?

  4th hexadecimal digit:
  Bit 3: Was "`Pa->house`" attribute used?
  Bit 2: Was "`Pa->street`" attribute used?
  Bit 1: Was "`Pa->lm`" attribute used?
  Bit 0: Was "`Pa->loc`" attribute used?

  5th hexadecimal digit:
  Bit 3: Was "`Pa->vtc`" attribute used?
  Bit 2: Was "`Pa->dist`" attribute used?
  Bit 1: Was "`Pa->state`" attribute used?
  Bit 0: Was "`Pa->pc`" attribute used?

  6th hexadecimal digit:
  Bit 3: Was "`Pfa->av`" attribute used?
  Bit 2: Was "`Pfa->lav`" attribute used?
  Bit 1: Was "`FMR`" used for biometric auth?
  Bit 0: Was "`FIR`" used for biometric auth?

  7th hexadecimal digit:
  Bit 3: Was "`IIR`" used for biometric auth?
  Bit 2: Was "`Pv->pin`" attribute used?
  Bit 1: Was "`Pv->otp`" attribute used?
  Bit 0: Was "`Tkn`" used?

8th hexadecimal digit:
Bit 3: Was "`Pa->po`" attribute used?
Bit 2: Was "`Pa->subdist`" attribute used?
Bit 1: Was "`Pi->dobt`" attribute used?
Bit 0: Was "SSK" used?

9th hexadecimal digit::
Bit 3: Was "Pi->name" attribute matched?
Bit 2: Was "Pi->lname" attribute matched?
Bit 1: Was "Pi->gender" attribute matched?
Bit 0: Was "Pi->dob" attribute matched?

10th hexadecimal digit::
Bit 3: Was "Pi->phone" attribute matched?
Bit 2: Was "Pi->email" attribute matched?
Bit 1: Was "Pi->age" attribute matched?
Bit 0: Was "Pa->co" attribute matched?

11th hexadecimal digit::
Bit 3: Was "Pa->house" attribute matched?
Bit 2: Was "Pa->street" attribute matched?
Bit 1: Was "Pa->lm" attribute matched?
Bit 0: Was "Pa->loc" attribute matched?

12th hexadecimal digit::
Bit 3: Was "Pa->vtc" attribute matched?
Bit 2: Was "Pa->dist" attribute matched?
Bit 1: Was "Pa->state" attribute matched?
Bit 0: Was "Pa->pc" attribute matched?

13th hexadecimal digit::
Bit 3: Was "Pfa->av" attribute matched?
Bit 2: Was "Pfa->lav" attribute matched?
Bit 1: Was "FMR/FIR" matched for biometric auth?
Bit 0: Was "IIR" matched for biometric auth?

14th hexadecimal digit:
Bit 3: Was "Pa->po" matched for biometric auth?
Bit 2: Was "Pa->subdist" attribute matched?
Bit 1: Was "Pi->dobt" attribute matched?
Bit 0: Was "Registered" device used?

15th hexadecimal digit:
Currently unused. Will have value 0.

- pid_version – Version string of the PID block which was part of input
- timestamp – PID "ts" string which was part of input
- fmrcount – Total number of FMR records which was part of input

- fircount – Total number of FIR records which was part of input
- iircount – Total number of IIR records which was part of input
- auth_api_ver – Version string of auth XML which was part of input
- SHA-256 of ASA code – Hash value of the ASA code
- SHA-256 of AUA code – hash value of the AUA code
- SUB-AUA code – This is same as "sc" attribute of input
- lot – Same as "lot" attribute value of "Meta" element of input
- lov – if "lot" is "G" (geo location), then this will be "lat|long|alt" where lat, long, and alt values are from input (separated by "|" char). If "lot" is "P" (postal pin code) then this will contain the "pincode" value
- lang – same as "lang" attribute of input. If input did not have it, this will contain value "NA".
- pi-ms – Match strategy used. Same as "ms" attribute of "Pi" element. If input did not have it, this will contain value "NA".
- pi-mv – Match value used. Same as "mv" attribute of "Pi" element. If input did not have it, this will contain value "NA".
- pi-lmv – Local language match value used. Same as "lmv" attribute of "Pi" element. If input did not have it, this will contain value "NA".
- pa-ms – Match strategy used. Same as "ms" attribute of "Pa" element. If input did not have it, this will contain value "NA".
- pa-lmv – Local language match value used. Same as "lmv" attribute of "Pa" element. If input did not have it, this will contain value "NA".
- pfa-ms – Match strategy used. Same as "ms" attribute of "Pfa" element. If input did not have it, this will contain value "NA".
- pfa-mv – Match value used. Same as "mv" attribute of "Pfa" element. If input did not have it, this will contain value "NA".
- pfa-lmv – Local language match value used. Same as "lmv" attribute of "Pfa" element. If input did not have it, this will contain value "NA".
- SHA-256 of tid – Hash value of "tid" attribute of input

  Example structure given below:

```
02{f676e1becb2e308770ac5212acbbc7d93ba5693d828714a5136b
6e1a9f438fc3,f25073ce9d46b0f720d00f32d8979c4efdab534686
8ffac90f4412d02710f7ef,0100002020000000,1.0,20130919122
506,1,0,0,1.6,1f5368b4cf6d7429033a47b8c7963329945c2bdf2
690fa3685945b15d3cda2e0,96cae35ce8a9b0244178bf28e4966c2
ce1b8385723a96a6b838858cdd6ca0a1e,SUBAUA0001,P,560103,N
A,P,50,NA,E,NA,NA,NA,NA,NA,591935b15b1c88e2d5f6be0a0546
04fcf36f0585a6f51098fa3803826fff278c}
```

Version 1.0 of Encoded Data:

<Version><SHA-256 of Aadhaar Number><SHA-256 of Demo element><Encoded Usage Data>

- "Version" – is the version of the info structure, "01"
- "SHA-256 of Aadhaar Number" – is the SHA-256 hash value of Aadhaar Number provided as part of input

- "SHA-256 of Demo Element" –
  - When Pid block is of type "X" (XML), then, it is the SHA-256 hash value of the substring representing Demo element in the Pid XML string.
  - When Pid block is of type "P" (Protobuf), then, it is the SHA-256 hash of the Protobuf value (byte array) of the Demo element.
- "Encoded Usage Data" - is 48 bit representation in HEX format of the various attribute usage of this authentication request.

  For version 1.0 of this block, Hexadecimal digits within the "Encoded Usage Data" should be interpreted based on below rules:

  1st hexadecimal digit:
  Bit 3-0: Version number of encoding. It will be hexadecimal "1" (binary: 0001) for encoding specified in this document.

  2nd hexadecimal digit:
  Bit 3: Was "`Pi->name`" attribute used?
  Bit 2: Was "`Pi->lname`" attribute used?
  Bit 1: Was "`Pi->gender`" attribute used?
  Bit 0: Was "`Pi->dob`" attribute used?

  3rd hexadecimal digit:
  Bit 3: Was "`Pi->phone`" attribute used?
  Bit 2: Was "`Pi->email`" attribute used?
  Bit 1: Was "`Pi->age`" attribute used?
  Bit 0: Was "`Pa->co`" attribute used?

  4th hexadecimal digit:
  Bit 3: Was "`Pa->house`" attribute used?
  Bit 2: Was "`Pa->street`" attribute used?
  Bit 1: Was "`Pa->lm`" attribute used?
  Bit 0: Was "`Pa->loc`" attribute used?

  5th hexadecimal digit:
  Bit 3: Was "`Pa->vtc`" attribute used?
  Bit 2: Was "`Pa->dist`" attribute used?
  Bit 1: Was "`Pa->state`" attribute used?
  Bit 0: Was "`Pa->pc`" attribute used?

  6th hexadecimal digit:
  Bit 3: Was "`Pfa->av`" attribute used?
  Bit 2: Was "`Pfa->lav`" attribute used?
  Bit 1: Was "`FMR`" used for biometric auth?
  Bit 0: Was "`FIR`" used for biometric auth?

  7th hexadecimal digit:
  Bit 3: Was "`IIR`" used for biometric auth?

Bit 2: Was "Pv->pin" attribute used?
Bit 1: Was "Pv->otp" attribute used?
Bit 0: Was "Tkn" used?

8th hexadecimal digit:
Bit 3: Was "Pa->po" attribute used?
Bit 2: Was "Pa->subdist" attribute used?
Bit 1: Was "Pi->dobt" attribute used?
Bit 0: Unused

9th to 12th Hexadecimal digits:
Currently unused. Will have value 0.

For example, if an authentication is done for Aadhaar Number "123412341234" and using demographic attributes "Name", "Gender", "DoB", "Phone", along with biometric FMR and OTP;

```
<Pid ts="2012-10-20T08:22:47" ver="1.0"><Demo><Pi
ms="E" name="Anand John" gender="M" dob="19690126"
phone="9999912345"/></Demo><Bios> <Bio type="FMR">
YjRmYmJkMTZkZGQ4OGQxYTY5YjI0M2ZiYjU4YTFlNmQwMmQ1YTgyYjN
mODU4YTMzYzQyZmNhOWUxN2QwNGVhNGMyMzExZjUyYmY4NjA5ZDVkZD
Y4YWU2NWE4OTNjNTMwNTJiM2U1YzQ5YTZkMGM2NzkyYTJlOGNhMTMxN
Dg0YWQ2MWM1ZGYzZGU0MTAzNzExZWVlN2E0MjU4ZjQ3ODg3NTU3ZWNm
YzcwY2NkM2QwZmIzZjg5OTg3NjEzNzA3ZDliZjkyMWU3NTc3OGU2NGJ
kMmM3MDhiNGQ4NDgyZGJmMGM3YjY3ZGZkZGZkNjIyNzgwMTllNjhhMm
I4MjQxZWY0MA==</Bio></Bios><Pv otp="111111"/></Pid>
```

then the value of the "info" attribute will be:
"0166c782e8f95ba958f28adaae576c42a263c2449af416fb844499
bef7fd41b2d00212be474bf2a6bfd4f361389ae66809babc1448291
29ae2315d6bab020455aa1B8002200000"
Where:

- "01" – is the version of info structure
- "66c782e8f95ba958f28adaae576c42a263c2449af416fb8444
99bef7fd41b2d0" – is the SHA-256 hash of Aadhaar Number
- "0212be474bf2a6bfd4f361389ae66809babc144829129ae231
5d6bab020455aa" – is the SHA-256 hash of "Demo" element (as a String)
- "1B8002200000" (binary bits "000110111000000000000010001000000000000000000000") – is the encoded usage data representing usage of Name, Gender, DoB, Phone, FMR, and OTP portions within authentication request

- **err** – Failure error code. If authentication fails ("ret" attribute value is "n"), this attribute provides any of the following codes (for latest updates on error codes, see https://developer.uidai.gov.in/site/api_err):
  - **"100"** – "Pi" (basic) attributes of demographic data did not match.
  - **"200"** – "Pa" (address) attributes of demographic data did not match
  - **"300"** – Biometric data did not match

- **"310"** – Duplicate fingers used
- **"311"** – Duplicate Irises used.
- **"312"** – FMR and FIR cannot be used in same transaction
- **"313"** – Single FIR record contains more than one finger
- "**314**" – Number of FMR/FIR should not exceed 10
- "**315**" – Number of IIR should not exceed 2
- "**400**" – Invalid OTP value
- "**401**" – Invalid TKN value
- **"500"** – Invalid encryption of Skey
- **"501"** – Invalid certificate identifier in "ci" attribute of "Skey"
- **"502"** – Invalid encryption of Pid
- **"503"** – Invalid encryption of Hmac
- **"504"** – Session key re-initiation required due to expiry or key out of sync
- **"505"** – Synchronized Key usage not allowed for the AUA
- **"510"** – Invalid Auth XML format
- **"511"** – Invalid PID XML format
- **"520"** – Invalid device
- **"521"** – Invalid FDC code under Meta tag
- **"522"** – Invalid IDC code under Meta tag
- **"530"** – Invalid authenticator code
- **"540"** – Invalid Auth XML version
- **"541"** – Invalid PID 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
- **"550"** – Invalid "Uses" element attributes
- **"551"** – Invalid "tid" value for registered device
- **"552"** – Invalid registered device key, please reset
- **"553"** – Invalid registered device HOTP, please reset
- **"554"** – Invalid registered device encryption
- **"555"** – Mandatory reset required for registered device
- **"561"** – Request expired ("Pid->ts" value is older than *N* hours where *N* is a configured threshold in authentication server)
- **"562"** – Timestamp value is future time (value specified "Pid->ts" is ahead of authentication server time beyond acceptable threshold)
- **"563"** – Duplicate request (this error occurs when exactly same authentication request was re-sent by AUA)
- **"564"** – HMAC Validation failed
- **"565"** – AUA license has expired
- **"566"** – Invalid non-decryptable license key
- **"567"** – Invalid input (this error occurs when some unsupported characters were found in Indian language values, "lname" or "lav")
- **"568"** – Unsupported Language
- **"569"** – Digital signature verification failed (means that authentication request XML was modified after it was signed)
- **"570"** – Invalid key info in digital signature (this means that certificate used for signing the authentication request is not valid – it is either

expired, or does not belong to the AUA or is not created by a well-known Certification Authority)

- **"571"** – PIN Requires reset (this error will be returned if resident is using the default PIN which needs to be reset before usage)
- "**572**" – Invalid biometric position
- **"573"** – Pi usage not allowed as per license
- **"574"**– Pa usage not allowed as per license
- **"575"**– Pfa usage not allowed as per license
- **"576"** - FMR usage not allowed as per license
- **"577"** – FIR usage not allowed as per license
- **"578"** – IIR usage not allowed as per license
- **"579"** – OTP usage not allowed as per license
- **"580"** – PIN usage not allowed as per license
- **"581"** – Fuzzy matching usage not allowed as per license
- **"582"** – Local language usage not allowed as per license
- **"584"** – Invalid pincode in LOV attribute under Meta tag
- **"585"** – Invalid geo-code in LOV attribute under Meta tag
- **"710"** – Missing "Pi" data as specified in "Uses"
- **"720"** – Missing "Pa" data as specified in "Uses"
- **"721"** – Missing "Pfa" data as specified in "Uses"
- **"730"** – Missing PIN data as specified in "Uses"
- **"740"** – Missing OTP data as specified in "Uses"
- **"800**" – Invalid biometric data
- **"810"** – Missing biometric data as specified in "Uses"
- **"811"** – Missing biometric data in CIDR for the given Aadhaar number
- **"812"** – Resident has not done "Best Finger Detection". Application should initiate BFD application to help resident identify their best fingers. See Aadhaar Best Finger Detection API specification.
- **"820"** – Missing or empty value for "bt" attribute in "Uses" element
- **"821"** – Invalid value in the "bt" attribute of "Uses" element
- **"901"** – No authentication data found in the request (this corresponds to a scenario wherein none of the auth data – Demo, Pv, or Bios – is present)
- **"902"** – Invalid "dob" value in the "Pi" element (this corresponds to a scenarios wherein "dob" attribute is not of the format "YYYY" or "YYYY-MM-DD", or the age of resident is not in valid range)
- **"910"** – Invalid "mv" value in the "Pi" element
- **"911"** – Invalid "mv" value in the "Pfa" element
- **"912"** – Invalid "ms" value
- **"913"** – Both "Pa" and "Pfa" are present in the authentication request (Pa and Pfa are mutually exclusive)
- **"930 to 939"** – Technical error that are internal to authentication server
- **"940"** – Unauthorized ASA channel
- **"941"** – Unspecified ASA channel
- **"980"** – Unsupported option
- **"997"** – Invalid Aadhaar status (Aadhaar is not in authenticatable status)
- **"998"** – Invalid Aadhaar Number
- **"999"** – Unknown error

4. API and Data Security

Broadly, Aadhaar authentication requests can be originated from either a "Registered" or a "Public" device. See registered Devices Specification for details of how they differ in working with biometric input.

4.1 Authentication Data Security

UIDAI provides a reference implementation of authentication client library for packaging and encrypting authentication data block in Java programming language. UIDAI may also provide other language implementations as necessary. Developers will be able to download these libraries along with UIDAI public key. Development community is encouraged to develop other programming language bindings and submit to UIDAI.

PID block data should be encrypted with a dynamic session key using **AES-256** symmetric algorithm (AES/ECB/PKCS7Padding). Session key, in turn, is encrypted with **2048-bit UIDAI public key** using asymmetric algorithm (RSA/ECB/PKCS1Padding). Reference implementation demonstrates this in detail. Session key must not be stored anywhere except in memory and should not be reused across transactions. Only re-use of session key that is allowed is its use as seed key when using synchronized session key scheme. To increase assurance multi-factor authentication using one-time pin (OTP) could also be used in conjunction with biometrics. See OTP API Specification for details on requesting OTP.

The encryption flow is as defined below:

1. Aadhaar Number, demographic, and biometric details as required by the application are entered into the device along with other factors such as OTP if it is used. If OTP is used, the request for OTP is sent to Aadhaar server along with Aadhaar Number (see "Aadhaar OTP Request API 1.5" specification). Aadhaar Authentication server sends the OTP back to the resident's registered mobile phone as an SMS and to the registered Email address.
2. AUA/Sub-AUA application generates a one-time session key.
3. The authentication "Data" XML block is encrypted using the one-time session key and then encoded (base 64).
4. The session key is then encrypted with the UIDAI public key.
5. AUA application on the device sends the encrypted block along with HMAC data to AUA server.
6. AUA server forms the final authentication XML input for API including license key, transaction reference ("txn" attribute), and sends the data to Aadhaar authentication server through an ASA network.
7. Aadhaar authentication server decrypts session key with the UIDAI private key. The data block is then decrypted using the session key.

8. The resident's decrypted biometric, demographic information, and optional OTP is taken into account during match based on the input.
9. Aadhaar authentication server responds with a "yes/no" as part of the digitally signed response XML.

4.2 Using Registered Devices

Authentication API supports using registered devices for biometric capture. Registered devices specification ensures biometrics data between device and host is encrypted and devices are uniquely identified.

From this API perspective, there are only two fundamental differences:

1. Biometric input must be obtained using "startCapture()" function which returns encrypted FMR/FIR/IIR block
    a. PID "ts" must be passed to this function as input
    b. This encrypted biometric record can directly be used within "bio" element of the PID block
2. Instead of using "public" as the "tid" value, when using registered device, "tid" must be obtained from the device by calling "getTid()" function

Details about the registered devices are available in specification document published at http://uidai.gov.in/images/aadhaar_registered_devices_1_0.pdf

4.3 Using Synchronized Session Key

As an advanced feature, this revision of the API version also supports a scheme called Synchronized Session Key. This allows session key to be sent in the beginning of the session and subsequent keys not to be sent for up to expiry of the session. At a high level, logic is as follows:

1. Authentication client generates an AES session key, generates a unique session identifier (key identifier) and sends the authentication request along with session key encrypted with PKI. Let's call it "seed" key.
2. For subsequent transactions within the maximum session expiry time (4 hours), authentication client does the following:
    a. Generate a new random number (CSPRNG) and generate a new AES key using the seed session key and this random number
    b. Send the authentication request along with the random number and key identifier without actually sending new session key
3. When seed session key expires (either 4 hours have passed since original session key was sent or server sends a 504 error), start with step 1.

4.4 Using Binary format for PID block

Protocol Buffers are a way of encoding structured data in an efficient yet extensible format. Google uses Protocol Buffers for almost all of its internal RPC protocols and file

formats. It provides a flexible, efficient, automated mechanism for serializing structured data. See https://developers.google.com/protocol-buffers/docs/overview for details.

This version of the API supports PID block to be sent to AUA server in protobuf format as an alternate to default XML format. This allows compact binary representation of the device data and avoids extra encoding required for XML format. If this scheme is used, device applications are expected to form the PID block in protobuf format using the ".proto" file for PID block. Final ".proto" files for this version are available at https://developer.uidai.gov.in/site/downloads

4.5 Authentication Audits

Aadhaar authentication will record all the authentication requests and their responses for audit purposes. By providing the Aadhaar number and authentication response code ("code" attribute in "AuthRes"), AUAs can request UIDAI to confirm the result of an authentication and authentication factors that were presented in that authentication request. UIDAI policy will determine how long these audits are maintained within CIDR.

All authentication responses are digitally signed by UIDAI and AUA's are recommended to validate the response integrity the keep track of these for audit purposes. In addition, attributes "ts", "info" within the API response can be used to verify if it the request was indeed for a particular Aadhaar number, if the request indeed had a biometric factor, when was the authentication done, etc. Such self verifiability of the authentication response allows 3rd party applications to trust and electronically verify the digitally signed response quite similar to that of an offline trust establishment against a gazetted officer signed paper.

5. Appendix

5.1 Related Publications

Demographic Data Standards	http://uidai.gov.in/UID_PDF/Committees/UID_DDSVP_Committee_Report_v1.0.pdf
Biometric Standards	http://uidai.gov.in/UID_PDF/Committees/Biometrics_Standards_Committee_report.pdf
Aadhaar biometric APIs	http://uidai.gov.in/UID_PDF/Working_Papers/Aadhaar_ABIS_API.pdf
Data Encryption Algorithm	ANXI X3.92
Banking—Retail Financial Services Symmetric Key Management	ANSI X9.24
Public Key Cryptography for the Financial Service Industry: Agreement of Symmetric Keys Using Discrete Cryptography	ANSI X9.42
Triple Data Encryption Algorithm: Modes of Operation	ANSI X9.52
Security Requirements for Cryptographic Modules	FIPS PUB 140-2
Personal Identification Number (PIN) Management and Security	ISO 9564
Information Technology – Security Techniques – Hash Functions	ISO 10118
Information Technology – Security Techniques – Key Management	ISO 11770
Information Technology – Security Techniques – Encryption Algorithms	ISO 18033
Biometric standards	ISO 19794-4, ISO 19794-6
Date and Time format standard	ISO_8601
XML Signature	http://www.w3.org/TR/xmldsig-core/
Indian e-governance standard for Metadata and Data standards for Person and Land Region codification	http://egovstandards.gov.in/standardsandFramework/metadata-and-data-standards/MDDS_Standard_release_version_1.0__Dec_24_2k9.pdf
Protocol Buffers	http://code.google.com/p/protobuf/
Geo Location Standard	ISO 6709

5.2 Changes in Version 1.6 from Version 1.5

Old (1.5)	New (1.6)
PID block is always XML	Added support for binary Protobuf format
"Txn" attribute was optional	"Txn" attribute made mandatory to ensure transactions are correlated and managed across systems well
Session Key is always sent	Allows advanced use case where session key is synchronized
Meta element was inside PID block	Moved "Meta" outside PID block to allow AUAs to validate them, log them, and use it in report
Meta element was optional	"Meta" element and its attributes made mandatory
--	"Meta" element structure changed to accommodate geo location, public IP address of the device, and a unique device code
--	Additional error codes added

Update in this document from previous release:

1. Txn ID naming convention is clarified further
2. Registered devices support and tid usage explained
3. Optional "actn" attribute added in response
4. "Info" attribute of response enhanced to v2
5. Additional error codes added