eSign API v1.0 - Specification

Table of Contents

Overview

- eSign is an online electronic integrated service that facilitates issuing a Digital Signature Certificate and performing signing of requested document/data.
- Electronic Signature is created using authentication of an individual through Aadhaar e- KYC service.
- Consent of the Aadhaar holder is obtained for Aadhaar authentication and eSign.
- Easy and secure way to digitally sign document anywhere, anytime.
- Facilitates legally valid signatures.
- Flexible and easy to implement.
- Privacy of the signer is maintained.
- Secure online service is used
- Immediate destruction of private keys after usage
- No hassles of key storage and key protection
- Saves cost and time
- User Convenience
- Legally recognized
- Suitable for individuals, businesses and Government
- Integrity with complete Audit trail
- No need of physical dongle

Process Flow

1. First make gateway modifications(refer Next section).
2. eSign can be done using either otp based authentication or biometric based authentication.
    a. For Otp based authentication, follow eSign request using otp section.
    b. For Biometric based authentication, follow eSign request using Biometrics section.

Required Libraries

- **AadhaarBridge-Gateway+application.properties**
  - Embedded jetty server
  - Acts as gateway of communication between AUA and SubAUA
- **AadhaarBridge_<x.y>.apk**
  - It is used to capture and encrypt biometric data
- **aadhaarbridge-aua-capture-wire-<x.y>.jar**
  - Contains classes for auth request/response & esign request/response
- **aadhaarbridge-esign-lib-<x.y>.jar**
  - Used for hash calculation and signature attachment to pdf

Download latest versions of above libraries from here.

Gateway Modification

1. Download the latest gateway.
2. Run the gateway jar using command:

```
$ java -jar aadhaar-gateway-<x.y.z>.jar
```

MAKE SURE THAT *application.properties* file contains **aua.host.esign.url**:

```
...
aua.host.auth.url=https://api.aadhaarbridge.com/aua/auth
aua.host.bfd.url=https://api.aadhaarbridge.com/aua/bfd
aua.host.otp.url=https://api.aadhaarbridge.com/aua/otp
aua.host.kyc.url=https://api.aadhaarbridge.com/aua/kyc
aua.host.mou.url=https://api.aadhaarbridge.com/aua/mou
aua.host.esign.url=https://api.aadhaarbridge.com/aua/esign
...
```

eSign Request using OTP

- **First generate otp:**

  - GateWay URL(endpoint: “/otp”, Method:POST)
    - When accessing using ip address http://<gateway public ip>:<port>/otp
    - When accessing using url *http://<Gateway URL>/otp*

  - *Json request for otp*

```
{
    "aadhaar-id": "<12 digit aadhaar number>",
    "certificate-type": "<preprod/prod>",
    "channel":"SMS/EMAIL",
    "type":"A"
}
```

  - *Otp request parameters*

Name	*Type*	*Value*	*Mandatory*	*description*
aadhaar-id	*String*	*12 digit aadhaar number*	*Yes*	*Aadhaar number of the resident who wants to eSign the document*
certificate-id	*String*	*Preprod or prod*	*Yes*	*It shows which environment you are using.*
channel	*String*	*SMS/EMAIL*	*Yes*	*If channel is SMS then otp would be generated on registred phone number with aadhaar or on registered email id if case of EMAIL*
type	*String*	*A*	*Yes*	

  - *Json response for otp*

```
{
        "success":true,
        "aadhaar-status-code":"",
        "aadhaar-reference-code":"”
}
```

➢ *Json response parameters*

Name	*Value*	*description*	*Example*
success	*true/false*	*This show the status of otp request*	
aadhaar-status-code	*Error code*	*In case of failure response, this error code will show the cause*	*111 i.e. aadhaar number does not have verified mobile*
aadhaar-reference-code	*Some random string*	*This code will be present in every response, if request reaches UIDAI server*	*8c5c631ab4dd69453 c1ef2e65ff3843c*

❖ ***After receiving otp on mobile/email, hit eSign request***

➢ GateWay URL(endpoint: "/esign/raw", Method:POST)
- When accessing using ip address *http://<gateway public ip>:<port>/esign/raw*
- When accessing using url *http://<Gateway URL>/esign/raw*

➢ *Json request for eSign*

```
{
"consent": "Y",
"response-sig-type":"pkcs7",
"docs":["hash for doc1,hash of doc2,...,hash of doc10"],
"auth-capture-request": {
        "aadhaar-id": "<12 digit aadhaar number>",
        "location": {
                "type": "pincode",
                "pincode": "<pincode of place from where eSign    request is
                generated>"
                },
        "modality": "otp",
        "certificate-type": "preprod" or "prod",
        "otp":"<6 digit otp>"
        }
}
```

- eSign Request Attributes

Parameter	*Type*	*Value*	*Mandatory*	*description*
consent	*String*	*“Y” or “N”*	*Yes*	*This is consent of aadhaar resident, eSign request can only be proceed with consent “Y”*
response-sig-type	*String*	*“pkcs7”*	*Yes*	*This is the type of signed hash*
docs	*Array of string*	*[“hash of doc1”,hash of doc2,..., hash of doc10]*	*Yes*	*This array contains the hash of documents which are to be signed. Maximum 10 documents’ hash can be send in one eSign request*
auth-capture-request	*Json*	*Otp based auth request*	*Yes*	*Otp Based Auth Request*

eSign Request using Biometrics

- Build and hit eSign request
    - GateWay URL(endpoint: “/esign”, Method:POST)
        - When accessing using ip address *http://<gateway public ip>:<port>/esign*
        - When accessing using url *http://<Gateway URL>/esign*

- *Json request for eSign*

```
{
        "consent": "Y",
        "response-sig-type":"pkcs7",
        "Docs":["sha-256 hash for doc1,sha-256 hash of doc2,...,sha-256 hash of doc10"],
        "auth-capture-data": {
                "modality": {
                        "fp-image": false/true,
                        "fp-minutae": false/true,
                        "iris": false/true
                       },
                "pid": {
                        "type": "xml/proto",
                        "value": "<encrypted pid data>"
                        },
                "aadhaar-id": "<12 digit aadhaar number>",
                "hmac": "<hmac value>",
                "location": {
                        "pincode": "<pincode of place from where request is generated>",
                        "type": "pincode"
                        },
                "public-ip": "<your public ip>",
                "session-key": {
                        "cert-id":"<expiry date of UIDAI public cert YYYYMMDD format>",
                        "value": "<session-key value>"
                        },
               "unique-device-code": "<your device code>"
                }
        }
```

➢ Json request parameters

Name	Type	Value	Mandatory	Description
consent	*String*	*“Y” or “N”*	*Yes*	*This is consent of aadhaar resident, eSign request can only be proceed with consent “Y”*
response-sig-type	*String*	*“pkcs7”*	*Yes*	*This is the type of signed hash*
docs	*List of string*	*[“sha-256 hash of doc1”,sha-256 hash of doc2,..., sha-256 hash of doc10]*	*Yes*	*This array contains the hash of documents which are to be signed. Maximum 10 documents’ hash can be send in one eSign request*
auth-capture-data	*Json*	*Auth request with biometric data*	*Yes*	*Biometric auth request*

eSign Response

❖ *Json response for esign*

```
{
        "success":true,
        "error-message":"<in case of failure>",
        "error-code":"<in case of failure>",
        "timestamp":"<ISO8601 format>",
        "reference-code":"<reference code>",
        "x509-cert":"public certificate of aadhaar resident",
        "auth-response":"",
        "doc-signatures":"<list of signed hash for all the documents>”
}
```

❖ eSign Response Parameters

Name	*Type*	*Value*	*description*	*Example*
success	*boolean*	*true or false*	*Status of esign request*	
error-message	*String*		*It shows the descriptive error message*	
error-code	*String*		*Error code which shows the cause of the error*	*See calculate document hash section*
timestamp	*String*		*Response timestamp in ISO format*	
reference-code	*String*		*Reference code will be unique for each transaction. It will be present for failure or success*	*Some random string*
x509-cert	*String*	*Base64 value*	*This contains the public certificate(.cer) of aadhaar resident*	
auth-response	*Json*		*Authentication response*	
doc-signatures	*List of String*	*Signed hash of all documents*	*This list contains signed hash of all document on base64 format*	

Auth Capture Data:

❖ Using our hidden apk(For android mobile only)
  ➢ Download latest apk.
  ➢ Install this apk on the mobile from where you want to capture the fingerprint data.
  ➢ List of supported devices by our apk are here.
  ➢ Follow authentication request and response section in android-sdk section.
  ➢ In response,you will get one json, set this json to auth-capture-data in eSign request with biometric.

- Using device’s sdk
  - If you are using some other devices or building app for other platforms then for building auth-capture-data, these biometrics related parameters will be taken from sdk
    - pid
    - hmac
    - session-key
    - Certificate id

  - Auth capture data parameters

Name	Type	Value	Mandatory	Description
fp-minutae	boolean	true/false	Yes	If fingerprint data is captured
iris	boolean	true/false	Yes	If eyes data is captured
pid-> type		xml/proto	Yes	xml, if encrypted biometric data is in xml format proto, if encrypted biometric data is in protobuf format
pid->value	String	Base64 string	Yes	This is the encrypted data in base64 format.This can be get using device sdk
aadhaar-id	String	12 digit aadhaar number	Yes	Aadhaar number of the resident who wants to sign the document
hmac	String	Base64 string	Yes	This can be get from device sdk
location->pincode	String	6 digit pincode	Yes	Pincode of the place from where eSign request is generated
location->type	String	pincode	Yes	
public-ip	String	Public ip of server	Yes	Ip of machine from where request is generated
session-key->cert-id	String	YYYYMMDD	Yes	It is the expiry date of uidai public

				certificate.This can be get from the device sdk
session-key->value	String	Base64 string	Yes	This can be get from device sdk
unique-device-code	String	Unique id of device	Yes	This can be get from device sdk

PDF Hash Calculation and Signing

E-Signing a document is slightly complex process. There are libraries like PdfBox, iText etc.. that are specifically created to handle pdfs programmatically. Since signer certificate is not available during e-signing, We have to follow external signing process which goes as follows:

1. Create signature field inside PDF with empty signature value.
2. Calculate Hash of pdf bytes including Signature field attributes but excluding Signature Value.
3. Once the SignedHashData comes back, we have to replace Signature Value with SignedHashData.

To make this hash calculation and signing process easy we have build SDK for java and android developers(see next section).

SDK for Java/Android developers

1. Download and Import **aadhaarbridge-esign-lib-1.0.jar**. This jar file contains "**SignatureData**" and "**ESignHelper**" classes.
2. Below code will help you generate Hash

```
SignatureData signatureData = new SignatureData();
signatureData.setLocation("Bangalore, Karnataka");
signatureData.setReason("Testing");
signatureData.setPageNumber(1);
signatureData.setSignatureName("mySignatureName");
signatureData.setRectangle(new Rectangle(50, 400, 200, 500));

InputStream is = new FileInputStream(“files/unsigned.pdf”);
OutputStream os = new FileOutputStream(“files/temp.pdf”);

ESignHelper helper = new ESignHelper();
String hash = helper.calculateHash(is, os, signatureData);
```

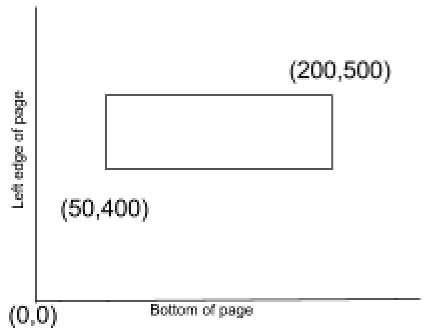
3. Hash generated in above step can be directly passed in Json request.
4. Construct Json request with OTP or Biometric and hit gateway(Refer Json above).
5. EsignResponse will contain respective signedData in “doc-signatures” array(Refer above).
6. Once you have signedData below code will help you attach signedData back to temporary file(that we created during hash generation code above).

```
InputStream is = new FileInputStream(“files/temp.pdf”);
OutputStream os = new FileOutputStream(“files/signed.pdf”);

ESignHelper helper = new ESignHelper();
helper.AttachSignature(is, os, signedData, signatureData.getSignatureName());
```

Points to Note:

1. signatureData in above code is same as you created before.(highlighted above).
2. *new Rectangle(50, 400, 200, 500).* ==> (x_1, y_1, x_2, y_2)



Error Codes:

❖ Additional Error Codes

Error Code	Description
110	Aadhaar number does not have verified mobile/email
111	Aadhaar number does not have verified mobile
112	Aadhaar number does not have both mobile and email
K-100	Resident authentication failed
ESP-905	Document hash not received