

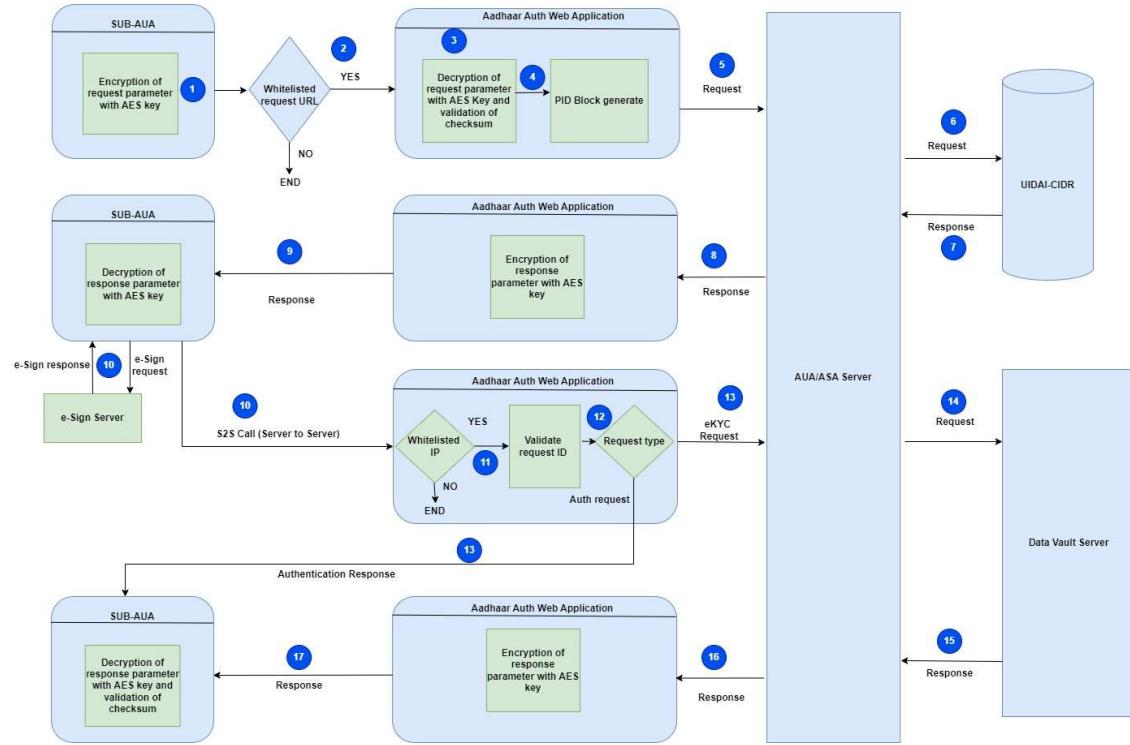


RAJ E-SIGN PROJECT,  
DEPARTMENT OF INFORMATION TECHNOLOGY &  
COMMUNICATION, GOR

**e-Sign API 1.1 INTEGRATION  
DOCUMENT  
(HTTP REDIRECTION MODE  
& MODAL POP-UP MODE)**

**Introduction:-** Presently, these Aadhaar authentication services are being provided through REST APIs. Now, AUA-DoIT&C is switching from API model to HTTP redirection based request-response system. This document describes the workflow and steps for integration with Aadhaar Authentication Application of AUA-DoIT&C using HTTP redirection and to perform eSign using eSign API v1.0.

### Work Flow Diagram- Aadhaar Authentication Framework



## **AES Key(Encryption Key) and AppCode**

### **AES Key**

Encryption key to encrypt the input JSON for Aadhaar Authentication Redirection process. Provide by Aadhaar Team.

### **AppCode**

To access the Aadhaar Authentication Redirection process. Provide by Aadhaar Team.

To generate AES Key and AppCode provide your department and application details in the given excel format to Aadhaar Team (Mail ids are provided in format) and download the format from given below link.

<https://rajca.rajasthan.gov.in/usermanual/AppCodeAndAESFormat.rar>

## **Transaction initiation**

### **1. HTTP REDIRECTION MODE**

This step defines the method to create the Aadhaar authentication POST request from the Web application of the client. Client is required to post the request parameters on *Request URL*.

#### **Request**

**UAT URL:** <https://aadhaarauthtest.rajasthan.gov.in/AadhaarAuth>

**Production URL:** <https://aadhaarauth.rajasthan.gov.in/AadhaarAuth>

**Method:** POST

Client application needs to post only two parameters named "AppCode" and "Data", where "AppCode" is in plain string and "Data" is the encrypted JSON string containing the Auth request details.

```
<form action=<AuthRequestEncrypted> method="POST">
<input type="hidden" name="AppCode" value=<AppCode>" />
<input type="hidden" name="Data" value=<Data>" />
<button type="submit">Submit</button>
</form>
```

- **Steps to create hidden filed "Data" value:**
- **Step 1:** Create the JSON string with request parameters as below
 

```
{
    "param1": "value1",
    "param2": "value2"
}
```

```

    "SSOToken": "<SSOToken >",
    "AppCode": "<AppCode >",
    "RequestId": "<RequestId >",
    "AadhaarNo": "<AadhaarNo >",
    "RequestType": "<RequestType >",
    "ReturnURL": "<ReturnURL>",
    "Checksum": "<Checksum >",
    "VerificationType": "<VerificationType >",
    "OperatorIdType": "<OperatorIdType >",
    "OperatorId": "<OperatorId >",
    "Purpose": "<Purpose >",

}

```

- **Step 2:** Encrypt the JSON string created in step 1 using AES key.

SR. No.	Request Parameter	Parameter Type	Parameter Constraints	Description
1	SSOToken	string		SSO session Token for SSO session validation
2	AppCode	string	Required Max Length: 32	Application-ID Issued by DoIT&C.
3	RequestId	String	Required Length: 18	This is the request id that is used to identify the particular request, it will be a unique ID and client will generate the request id. RequestId format should be as below: - <i>Sub-AUA Code + 8-digit Random number.</i> For Ex.- PRXXXXXXX + 12345678
4	AadhaarNo	String	Required Min Length: 12 Max Length: 15	12 Digit Aadhaar Number/15 Digit Aadhaar Reference No./16 digit VID of beneficiary whose Aadhaar Face Authentication is to be performed. Encrypted using AES key.
5	RequestType	Int	Required	This parameter defines type of response required by the client application after face authentication. It accepts only two integer values- 1 - for Yes/No response 2 - for eKYC 3 – for eSign
6	ReturnURL	String	Required	It contains the return URL of the client application. It will be used for return to Client application after Face Authentication. This needs to be

				whitelisted at Aadhaar Authentication application.								
<b>7</b>	Checksum	Hash		fixed checksum hash of values AppCode, RequestId, AadhaarNo, RequestType								
<b>8</b>	VerificationType	Int	Required for Face Authentication	Verification Type parameter defines user type who will perform Face Authentication. It accepts only two integer values as below: - 1 - Beneficiary 2 - Authentication Operator								
<b>9</b>	OperatorIdType	Int	Required if Face Authentication is to be performed through an Authentication Operator	This defines the ID Type of Authentication operator who will perform Face Authentication in his mobile. This field accepts only three integer values- <table border="1" data-bbox="971 686 1387 982"> <thead> <tr> <th>OperatorIdType</th><th>OperatorId</th></tr> </thead> <tbody> <tr> <td>1</td><td>SSO ID</td></tr> <tr> <td>2</td><td>12 Digit Aadhaar Number</td></tr> <tr> <td>3</td><td>15 Digit Aadhaar Reference No.</td></tr> </tbody> </table>	OperatorIdType	OperatorId	1	SSO ID	2	12 Digit Aadhaar Number	3	15 Digit Aadhaar Reference No.
OperatorIdType	OperatorId											
1	SSO ID											
2	12 Digit Aadhaar Number											
3	15 Digit Aadhaar Reference No.											
<b>10</b>	OperatorId	String	Required if Face Authentication is to be performed through a Authentication Operator	This is the id of Authentication operator who will perform Face Authentication in his mobile. This field accepts only three string values based on OperatorIdType as below – <table border="1" data-bbox="971 1203 1387 1499"> <thead> <tr> <th>OperatorIdType</th><th>OperatorId</th></tr> </thead> <tbody> <tr> <td>1</td><td>SSO ID</td></tr> <tr> <td>2</td><td>12 Digit Aadhaar Number</td></tr> <tr> <td>3</td><td>15 Digit Aadhaar Reference No.</td></tr> </tbody> </table>	OperatorIdType	OperatorId	1	SSO ID	2	12 Digit Aadhaar Number	3	15 Digit Aadhaar Reference No.
OperatorIdType	OperatorId											
1	SSO ID											
2	12 Digit Aadhaar Number											
3	15 Digit Aadhaar Reference No.											
<b>11</b>	Purpose	String	Required Max length: 100	This parameter defines the purpose of the Aadhaar Authentication Authentication i.e. name of any schemes or any other purpose.								

After Creation of the POST request, client application will be redirected to Aadhaar Authentication application page.

आधार प्रमाणन/ ई-केवाईसी

2:13

हिन्दी Eng

| विभाग  
DOIT & C

आधार नंबर  
XXXXXXX-3800 | अनुरोध आईडी  
XXXXXXXXXXXXXX-3178 | अनुरोध प्रकार  
Authentication | अनुरोध समय  
06-08-2024 10:11 AM

उद्देश्य : Face Auth

 ओटीपी/ टीओटीपी

 चेहरा प्रमाणीकरण

 अंगूली की छाप

 आँख की पुतली

मैं एतद्वारा घोषणा करता/ करती हूँ कि मुझे DOIT & C के साथ Face Auth हेतु आधार आधारित प्रमाणीकरण प्रणाली के साथ अपने आप को प्रमाणित करने में कोई आपत्ति नहीं है और मैं आधार आधारित हाँ/ना के लिए अपने आधार संख्या, बैंयोमीट्रिक और/ या ओटीपी प्रदान करने के लिए सहमति देता/ देती हूँ। आधार सिस्टम से मोबाइल नंबर और ईमेल तक पहुंचने के लिए मैं अपनी स्पष्ट सहमति भी देता/ देती हूँ।

मेरे पास पहले से एक वैध आधार ओटीपी/ टीओटीपी है।



ओटीपी भेजे रद्द करें

मोबाइल नंबर अपडेट होना चाहिए। आधार में मोबाइल नंबर अपडेट करने के लिए कृपया अपने निकटतम आधार नामांकन

- Aadhaar holder needs to select the authentication mode i.e. OTP, Fingerprint, Iris, PI/PA and face Auth etc. and perform authentication as per requirement.

## 2. MODAL POP-UP MODE

To create an Aadhaar authentication request from the client's web application, follow these steps:

- **Include the Aadhaar Authentication Script:** Add the JavaScript file to your HTML by including the following <script> tag in your web application:

### JavaScript CDN:

(UAT)<scriptsrc="https://aadhaarauthtest.rajasthan.gov.in/service/aadhaarauth.js"></script>

(PROD.)<scriptsrc="https://aadhaarauth.rajasthan.gov.in/service/aadhaarauth.js"></script>

- **Initialize and Configure Authentication:** The aadhaarauth.js script provides necessary functions and methods to perform authentication. Once included, initialize and configure the authentication request with required parameters
- Client application needs to pass only two parameters named "AppCode" and "Data", where "AppCode" is in plain string and "Data" is the encrypted JSON string containing the authentication request details.
- Using the methods available in the aadhaarauth script, build a function in JavaScript that authenticate the user's Aadhaar details.

```
const authInstance = new window.AadhaarAuth({ AppCode: "", Data: "" });
authInstance.open({
  handler: (response) => {
    console.log("Authentication successful", response);
    // Handle successful authentication
  }
});
```

- **Steps to hidden filed “Data” value:**

- **Step 1:** Create the JSON string with request parameters as below

```
{
  "SSOToken": "<SSOToken >",
  "AppCode": "<AppCode >",
  "RequestId": "<RequestId >",
  "AadhaarNo": "<AadhaarNo >",
  "RequestType": "<RequestType >",
  "ReturnURL": "<ReturnURL>",
  "Checksum": "<Checksum >",
  "PiAuth": "{\"Name": "", \"LastName\": \"\", \"Dob\": \"\", \"Gender\": \"\", \"Age\": \"\", \"Phone\": \"\", \"Email\": \"\"}",
  "PaAuth": "{\"Co\": \"\", \"House\": \"\", \"Street\": \"\", \"Lm\": \"\", \"Loc\": \"\", \"Vtc\": \"\", \"Subdist\": \"\", \"Dist\": \"\",
  \"Pc\": \"\", \"Po\": \"\"}",
  "VerificationType": "<VerificationType >",
  "OperatorIdType": "<OperatorIdType >",
  "OperatorId": "<OperatorId >",
  "Purpose": "<Purpose >",
  "AuthModality": "<AuthModality >",
}
```

- **Step 2:** Encrypt the JSON string created in step 1 using AES key.

SR. No.	Request Parameter	Parameter Type	Parameter Constraints	Description
1	SSOToken	string		SSO session Token for SSO session validation
2	AppCode	string	Required Max Length: 32	Application-ID Issued by DoIT&C.
3	RequestId	String	Required Length: 18	This is the request id that is used to identify the particular request, it will be a unique ID and client will generate the request id. RequestId format should be as below: - <i>Sub-AUA Code + 8-digit Random number.</i> For Ex.- PRXXXXXXX + 12345678
4	AadhaarNo	String	Required Min Length: 12 Max Length: 15	12 Digit Aadhaar Number/15 Digit Aadhaar Reference No./16 digit VID of beneficiary whose Aadhaar Face Authentication is to be performed. Encrypted using AES key.
5	RequestType	Int	Required	This parameter defines type of response required by the client application after face authentication. It accepts only two integer values- 1 - for Yes/No response 2 - for eKYC 3 – for eSign
6	ReturnURL	String	Required	It contains the return URL of the client application. It will be used for return to Client application after Face Authentication. This needs to be whitelisted at Aadhaar Authentication application.
7	Checksum	Hash		fixed checksum hash of values AppCode, RequestId, AadhaarNo, RequestType
8	PiAuth	json	Optional	Accepts parameters in json format for performing Personal Identity (PI) authentication of Aadhaar holders
9	PiAuth	json	Optional	Accepts parameters in json format for performing Address Identity (PA) authentication of Aadhaar holders
10	VerificationType	Int	Required for Face Authentication	Verification Type parameter defines user type who will perform Face Authentication. It accepts only two integer values as below: - 1 - Beneficiary 2 - Authentication Operator

<b>11</b>	OperatorIdType	Int	Required if Face Authentication is to be performed through an Authentication Operator	This defines the ID Type of Authentication operator who will perform Face Authentication in his mobile. This field accepts only three integer values-								
				<table border="1"> <thead> <tr> <th>OperatorIdType</th><th>OperatorId</th></tr> </thead> <tbody> <tr> <td>1</td><td>SSO ID</td></tr> <tr> <td>2</td><td>12 Digit Aadhaar Number</td></tr> <tr> <td>3</td><td>15 Digit Aadhaar Reference No.</td></tr> </tbody> </table>	OperatorIdType	OperatorId	1	SSO ID	2	12 Digit Aadhaar Number	3	15 Digit Aadhaar Reference No.
OperatorIdType	OperatorId											
1	SSO ID											
2	12 Digit Aadhaar Number											
3	15 Digit Aadhaar Reference No.											
<b>12</b>	OperatorId	String	Required if Face Authentication is to be performed through a Authentication Operator	This is the id of Authentication operator who will perform Face Authentication in his mobile. This field accepts only three string values based on OperatorIdType as below –								
				<table border="1"> <thead> <tr> <th>OperatorIdType</th><th>OperatorId</th></tr> </thead> <tbody> <tr> <td>1</td><td>SSO ID</td></tr> <tr> <td>2</td><td>12 Digit Aadhaar Number</td></tr> <tr> <td>3</td><td>15 Digit Aadhaar Reference No.</td></tr> </tbody> </table>	OperatorIdType	OperatorId	1	SSO ID	2	12 Digit Aadhaar Number	3	15 Digit Aadhaar Reference No.
OperatorIdType	OperatorId											
1	SSO ID											
2	12 Digit Aadhaar Number											
3	15 Digit Aadhaar Reference No.											
<b>13</b>	Purpose	String	Required Max length: 100	This parameter defines the purpose of the Aadhaar Authentication Authentication i.e. name of any schemes or any other purpose.								
<b>14</b>	AuthModality	String	Optional	This parameters is to be used to decide the authentication mode. "0": All Modes "1": OTP "2": Face "3": Finger "4": Iris "5": PI/PA								

विभाग

DOIT &amp; C

आधार नंबर

XXXXXXXXXX-3800

उद्देश्य : Face Auth

अनुरोध आईडी

XXXXXXXXXXXXXX-3178

अनुरोध प्रकार

Authentication

अनुरोध समय

06-08-2024 10:11 AM



ओटीपी/ टीओटीपी



चेहरा प्रमाणीकरण



अंगूली की छाप



आँख की पुतली

मैं एतद्वारा घोषणा करता/ करती हूँ कि मुझे DOIT & C के साथ Face Auth हेतु आधार आधारित प्रमाणीकरण प्रणाली के साथ अपने आप को प्रमाणित करने में कोई आपत्ति नहीं है और मैं आधार आधारित हाँ/ना के लिए अपने आधार संख्या, बॉयोमेट्रिक और/ या ओटीपी प्रदान करने के लिए सहमति देता/ देती हूँ। आधार सिस्टम से मोबाइल नंबर और ईमेल तक पहुँचने के लिए मैं अपनी स्पष्ट सहमति भी देता/ देती हूँ।

मेरे पास पहले से एक वैध आधार ओटीपी/ टीओटीपी है।

ओटीपी भेजे

रद्द करें



मोबाइल नंबर अपडेट होना चाहिए। आधार में मोबाइल नंबर अपडेट करने के लिए कृपया अपने निकटतम आधार नामांकन

- Aadhaar holder needs to select the authentication mode i.e. OTP, Fingerprint, Iris, PI/PA and face Auth etc. and perform authentication as per requirement.

## Transaction Response

After execution of Aadhaar Authentication request, response is provided via POST method on the *ReturnURL* in JSON format. Response JSON will contain one parameters only “Data”, where “Data” is the encrypted JSON string containing the Aadhaar Auth response details.

**Method:** POST

**Content-Type:** application/json

**Response json:**

```
{
  "Data": ""
}
```

“Data” parameter contains following JSON string in encrypted format i.e. Sub-AUA application will get following JSON after decryption of “Data” at client end:-

```
{
  "RequestId": "RequestId",
  "Message": "Your request has been executed successfully",
  "ResponseCode": "200"
}
```

## **Server to Server API call for digital signature on document For Single Mode(eSign API 1.0)**

### **Request**

#### **UAT URL:**

[https://apitest.sewadwaar.rajasthan.gov.in/app/live/rajesign/Staging/Service/all/webresources/generic/esign/Doc?client\\_id=](https://apitest.sewadwaar.rajasthan.gov.in/app/live/rajesign/Staging/Service/all/webresources/generic/esign/Doc?client_id=)

#### **PRODUCTION URL:**

[https://api.sewadwaar.rajasthan.gov.in/app/live/rajesign/Prod/Service/all/webresources/generic/esign/Doc?client\\_id=](https://api.sewadwaar.rajasthan.gov.in/app/live/rajesign/Prod/Service/all/webresources/generic/esign/Doc?client_id=)

#### **Method: POST**

**Content-Type:** application/json

**Request json:** Request from e-Sign

```
{  
    "inputJson": {  
        "File": ""  
    },  
    "filetype": "PDF",  
    "transactionid": "Request ID generated for Aadhaar authentication",  
    "docname": "xxx",  
    "designation": "xxx",  
    "status": "",  
    "llx": "",  
    "lly": "",  
    "positionX": "",  
    "positionY": "",  
    "mode": "",  
    "page": ""  
}
```

### **Response**

**Method:** POST

**Content-Type:** application/json

**Response json:**

**Successful Response json:**

```
{  
    "Status": "1",  
    "ResponseCode": "MS-111",  
    "SuccessMessage": "Esign Has been done SUccessfully",  
    "Document": "digitally signed base-64 string",  
    "Transaction Id": "Request ID generated for Aadhaar authentication"  
}
```

**Failed Response json:**

```
{  
    "Status": "0",  
    "ResponseCode": "",  
    "Transaction Id": "Request ID generated for Aadhaar authentication",  
    "ErrorMessage": ""  
}
```

## **Server to Server API call for digital signature on document For Generic eSignDocement(eSign API 1.0)**

### **Request**

#### **UAT URL:**

[https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/UAT/webresources/generic/eSignedocument?client\\_id=](https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/UAT/webresources/generic/eSignedocument?client_id=)

#### **PRODUCTION URL:**

[https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/webresources/Service/generic/eSignedocument?client\\_id=](https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/webresources/Service/generic/eSignedocument?client_id=)

#### **Method: POST**

**Content-Type:** application/json

**Request json:** Request from e-Sign

```
{  
    "inputJson": {  
        "File": ""  
    },  
    "filetype": "",  
    "transactionid": "Request ID generated for Aadhaar authentication",  
    "docname": "xxx",  
    "designation": "xxx",  
    "status": "",  
    "llx": "",  
    "lly": "",  
    "positionX": "",  
    "positionY": "",  
    "page": ""  
}
```

### **Response**

**Method:** POST

**Content-Type:** application/json

**Response json:**

**Successful Response json:**

```
{  
    "Status": "1",  
    "ResponseCode": "MS-111",  
    "SuccessMessage": "Esign has been done Successfully",  
    "Document": "digitally signed base-64 string",  
    "Transaction Id": "Request ID generated for Aadhaar authentication"  
}
```

**Failed Response json:**

```
{  
    "Status": "0",  
    "ResponseCode": "",  
    "Transaction Id": "Request ID generated for Aadhaar authentication",  
    "ErrorMessage": ""  
}
```

## **Server to Server API call for digital signature on document For Generic MultipleEsignPage (eSign API 1.0)**

### **Request**

#### **UAT URL:**

[https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/UAT/webresources/generic/multipleEsignPage?client\\_id=](https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/UAT/webresources/generic/multipleEsignPage?client_id=)

#### **PRODUCTION URL:**

[https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/generic/webresources/multiple/eSignPage?client\\_id=](https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/generic/webresources/multiple/eSignPage?client_id=)

#### **Method: POST**

**Content-Type:** application/json

**Request json:** Request from e-Sign

```
{  
    "inputJson": {  
        "File": ""  
    },  
    "filetype": "",  
    "transactionid": "Request ID generated for Aadhaar authentication",  
    "docname": "xxx",  
    "designation": "xxx",  
    "status": "",  
    "llx": "",  
    "lly": "",  
    "positionX": "",  
    "positionY": "",  
    "page": ""  
}
```

### **Response**

**Method:** POST

**Content-Type:** application/json

**Response json:**

**Successful Response json:**

```
{  
    "Status": "1",  
    "ResponseCode": "MS-111",  
    "SuccessMessage": "Esign Has been done Successfully",  
    "Document": "digitally signed base-64 string",  
    "Transaction Id": "Request ID generated for Aadhaar authentication"  
}
```

**Failed Response json:**

```
{  
    "Status": "0",  
    "ResponseCode": "",  
    "Transaction Id": "Request ID generated for Aadhaar authentication",  
    "ErrorMessage": ""  
}
```

## **Server to Server API call for digital signature on document For Generic MultipleEsignedDocument (eSign API 1.0)**

### **Request**

#### **UAT URL:**

[https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/UAT/webresources/generic/multipleESigneddocument?client\\_id=](https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/UAT/webresources/generic/multipleESigneddocument?client_id=)

#### **PRODUCTION URL:**

[https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/webresources/multiple/generic/eSignedocument ?client\\_id=](https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/webresources/multiple/generic/eSignedocument ?client_id=)

#### **Method: POST**

**Content-Type:** application/json

**Request json:** Request from e-Sign

```
{  
    "inputJson": {  
        "File": ""  
    },  
    "filetype": "",  
    "transactionid": "Request ID generated for Aadhaar authentication",  
    "docname": "xxx",  
    "designation": "xxx",  
    "status": "",  
    "llx": "",  
    "lly": "",  
    "positionX": "",  
    "positionY": "",  
    "page": ""  
}
```

### **Response**

**Method:** POST

**Content-Type:** application/json

**Response json:**

**Successful Response json:**

```
{  
    "Status": "1",  
    "ResponseCode": "MS-111",  
    "SuccessMessage": "Esign Has been done SUccessfully",  
    "Document": "digitally signed base-64 string",  
    "Transaction Id": "Request ID generated for Aadhaar authentication"  
}
```

**Failed Response json:**

```
{  
    "Status": "0",  
    "ResponseCode": "",  
    "Transaction Id": "Request ID generated for Aadhaar authentication",  
    "ErrorMessage": ""  
}
```

## **Server to Server API call for digital signature on document For Generic PagewiseEsignedDocument(eSign API 1.0)**

### **Request**

#### **UAT URL:**

[https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/UAT/webresources/generic/pageWiseESignedocument?client\\_id=](https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/UAT/webresources/generic/pageWiseESignedocument?client_id=)

#### **PRODUCTION URL:**

[https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/webresources/specific/pageWise/generic/eSignedocument?client\\_id=](https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/webresources/specific/pageWise/generic/eSignedocument?client_id=)

#### **Method: POST**

**Content-Type:** application/json

**Request json:** Request from e-Sign

```
{  
    "inputJson": {  
        "File": ""  
    },  
    "filetype": "",  
    "transactionid": "Request ID generated for Aadhaar authentication",  
    "docname": "xxx",  
    "designation": "xxx",  
    "status": "",  
    "llx": "",  
    "lly": "",  
    "positionX": "",  
    "positionY": "",  
    "page": ""  
}
```

### **Response**

**Method:** POST

**Content-Type:** application/json

**Response json:**

**Successful Response json:**

```
{  
    "Status": "1",  
    "ResponseCode": "MS-111",  
    "SuccessMessage": "Esign Has been done SUccessfully",  
    "Document": "digitally signed base-64 string",  
    "Transaction Id": "Request ID generated for Aadhaar authentication"  
}
```

**Failed Response json:**

```
{  
    "Status": "0",  
    "ResponseCode": "",  
    "Transaction Id": "Request ID generated for Aadhaar authentication",  
    "ErrorMessage": ""  
}
```

## **Server to Server API call for digital signature on document For Bulk EsignedDocument(eSign API 1.0)**

### **Request**

#### **UAT URL:**

[https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Staging/bulkesign/webresources/generic/bulk/document?client\\_id=](https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Staging/bulkesign/webresources/generic/bulk/document?client_id=)

#### **PRODUCTION URL:**

[https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/bulkesign/webresources/generic/bulk/document?client\\_id=](https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/bulkesign/webresources/generic/bulk/document?client_id=)

#### **Method: POST**

**Content-Type:** application/json

**Request json:** Request from e-Sign

```
{  
    "inputJson": {  
        "File": ""  
    },  
    "filetype": "",  
    "transactionid": "Request ID generated for Aadhaar authentication",  
    "docname": "xxx",  
    "designation": "xxx",  
    "status": "",  
    "llx": "",  
    "lly": "",  
    "positionX": "",  
    "positionY": "",  
    "page": ""  
}
```

### **Response**

**Method:** POST

**Content-Type:** application/json

**Response json:**

**Successful Response json:**

```
{  
    "Status": "1",  
    "ResponseCode": "MS-111",  
    "SuccessMessage": "Esign Has been done SUccessfully",  
    "Document": "digitally signed base-64 string",  
    "Transaction Id": "Request ID generated for Aadhaar authentication"  
}
```

**Failed Response json:**

```
{  
    "Status": "0",  
    "ResponseCode": "",  
    "Transaction Id": "Request ID generated for Aadhaar authentication",  
    "ErrorMessage": ""  
}
```

## **Server to Server API call for digital signature on document For Bulk MultipleEsignPage (eSign API 1.0)**

### **Request**

#### **UAT URL:**

[https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Staging/bulkesign/webresources/generic/multiple/page?client\\_id=](https://apitest.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Staging/bulkesign/webresources/generic/multiple/page?client_id=)

#### **PRODUCTION URL:**

[https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/bulkesign/webresources/generic/multiple/page?client\\_id=](https://api.sewadwaar.rajasthan.gov.in/app/live/RajeSign/Prod/bulkesign/webresources/generic/multiple/page?client_id=)

#### **Method: POST**

**Content-Type:** application/json

**Request json:** Request from e-Sign

```
{  
    "inputJson": {  
        "File": ""  
    },  
    "filetype": "",  
    "transactionid": "Request ID generated for Aadhaar authentication",  
    "docname": "xxx",  
    "designation": "xxx",  
    "status": "",  
    "llx": "",  
    "lly": "",  
    "positionX": "",  
    "positionY": "",  
    "page": ""  
}
```

### **Response**

**Method:** POST

**Content-Type:** application/json

**Response json:**

**Successful Response json:**

```
{  
    "Status": "1",  
    "ResponseCode": "MS-111",  
    "SuccessMessage": "Esign Has been done Successfully",  
    "Document": "digitally signed base-64 string",  
    "Transaction Id": "Request ID generated for Aadhaar authentication"  
}
```

**Failed Response json:**

```
{  
    "Status": "0",  
    "ResponseCode": "",  
    "Transaction Id": "Request ID generated for Aadhaar authentication",  
    "ErrorMessage": ""  
}
```

## Details of request and response Parameters for e-Sign

SR. No.	Request/Response Parameter	Parameter Type	Description
1	File	String	It contains the BASE 64 of file
2	filetype	String	It's a type of file and it can be a PDF or XML/excel
3	transactionid	String	Request ID generated for Aadhaar authentication.
4	docname	String	docname provided by department
5	designation	String	Designation of employee
6	status	String	Status of e-sign returned by e-Sign application (Approved/SelfAttested)
7	lx	String	X- coordinate of sign
8	ly	String	Y-coordinate of sign
9	positionX	String	X- coordinate of QR code
10	positionY	String	Y-coordinate of QR code
11	Mode(For single Mode url only)	String	Mode parameter defines, on which page you want to sign, which is 1: e-sign on the last page of the document 2: Multiple users can sign on the document 3: e-sign on every page of the document 4: e-sign on particular page
12	page	String	Page No. you want to sign.
13	ResponseCode	String	Response code related to e-sign application
14	SuccessMessage	String	This is used to show the status or any error message returned by e-sign application.
15	Document	String BASE-64	Signed Document

## Sample Code for encryption and decryption:-

### ASP .Net

#### -----Encryption-----

```
public static EncryptionResult Encrypt(string plainText, byte[] key)
{
    EncryptionResult model = new();
    try
    {
        using Aes aesAlg = Aes.Create();
        aesAlg.Key = key;
        aesAlg.GenerateIV();

        ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);
        using MemoryStream msEncrypt = new();
        msEncrypt.Write(aesAlg.IV, 0, aesAlg.IV.Length);

        using (CryptoStream csEncrypt = new(msEncrypt, encryptor, CryptoStreamMode.Write))
        {
            using StreamWriter swEncrypt = new(csEncrypt);
            swEncrypt.WriteLine(plainText);
        }
        model.IsSuccess = true;
        model.EncryptedData = msEncrypt.ToArray();
        return model;
    }
    catch (Exception ex)
    {
        model.IsSuccess = false;
        model.Message = ex.Message;
        return model;
    }
}
```

---

### -----Decryption-----

```
private static EncryptionResult DecryptWithKey(byte[] cipherText, string keyString)
{
    EncryptionResult model = new();
    byte[] key = Encoding.UTF8.GetBytes(keyString);
    try
    {
        using Aes aesAlg = Aes.Create();

        aesAlg.Key = key;
        byte[] iv = new byte[aesAlg.IV.Length];
        Array.Copy(cipherText, 0, iv, 0, iv.Length);
        aesAlg.IV = iv;
        ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);
        using MemoryStream msDecrypt = new(cipherText, iv.Length, cipherText.Length - iv.Length);
        using CryptoStream csDecrypt = new(msDecrypt, decryptor, CryptoStreamMode.Read);
        using StreamReader srDecrypt = new(csDecrypt);
        model.IsSuccess = true;
        model.DecryptedData = srDecrypt.ReadToEnd();
    }
    catch (Exception ex)
    {
        model.IsSuccess = false;
        model.Message = ex.Message;
    }
    return model;
}
```

---

**Class**

---

```
public class EncryptionResult{  
    public bool IsSuccess { get; set; }  
    public byte[] EncryptedData { get; set; } = new byte[0];  
    public string DecryptedData { get; set; } = string.Empty;  
    public string Message { get; set; } = string.Empty;  
}
```

---

**Method**

---

```
byte[] encData = Convert.FromBase64String('JSON STRING');  
var requestData = Utilities.Decrypt(encData, Key);
```

---

**Generate Checksum**

---

```
public static string ComputeSHA256Hash(string input)  
{  
    using SHA256 sha256 = SHA256.Create();  
    byte[] inputBytes = Encoding.UTF8.GetBytes(input);  
    byte[] hashBytes = sha256.ComputeHash(inputBytes);  
    StringBuilder hex = new(hashBytes.Length * 2);  
    foreach (byte b in hashBytes)  
    {  
        hex.AppendFormat("{0:x2}", b);  
    }  
    return hex.ToString();  
}
```

## Java

### -----Encryption-----

```
public static byte[] encrypt(String plainText, byte[] key) throws Exception {  
    if (plainText == null || key == null)  
    {  
        throw new IllegalArgumentException("Plain text or key cannot be null");  
    }  
  
    //Create AES cipher  
  
    Cipher aesCipher = Cipher.getInstance("AES/CBC/PKCS5Padding");  
    //Create secret key  
  
    SecretKeySpec secretKeySpec = new SecretKeySpec(key, "AES");  
    //Generate a random IV (Initialization Vector)  
  
    byte[] iv = new byte[16];  
    SecureRandom secureRandom = new SecureRandom();  
    secureRandom.nextBytes(iv);  
    IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);  
  
    // Initialize cipher for encryption  
  
    aesCipher.init(Cipher.ENCRYPT_MODE, secretKeySpec, ivParameterSpec);  
    //Perform encryption  
  
    byte[] encrypted;  
    try (ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream()) {  
        //Write IV to the output stream  
  
        byteArrayOutputStream.write(iv);  
  
        //Encrypt plaintext  
  
        byte[] encryptedData = aesCipher.doFinal(plainText.getBytes(StandardCharsets.UTF_8));  
        byteArrayOutputStream.write(encryptedData);  
        encrypted = byteArrayOutputStream.toByteArray();  
    }  
    catch (IOException e) {  
        throw new RuntimeException("Error encrypting data", e);  
    }  
  
    return encrypted;  
}
```

-----Decryption-----

```
public static String decrypt(byte[] encryptedData, byte[] key)
throws Exception {if (encryptedData == null || key == null) {
throw new IllegalArgumentException("Encrypted data or key cannot be null");
}
//Create AES cipher

Cipher aesCipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
//Extract IV from the encrypted data
byte[] iv = Arrays.copyOfRange(encryptedData, 0,
16); IvParameterSpec ivParameterSpec = new
IvParameterSpec(iv);
//Create secret key

SecretKeySpec secretKeySpec = new SecretKeySpec(key, "AES");
//Initialize cipher for decryption

aesCipher.init(Cipher.DECRYPT_MODE, secretKeySpec, ivParameterSpec);
//Perform decryption

byte[] decryptedData =
aesCipher.doFinal(Arrays.copyOfRange(encryptedData, 16,
encryptedData.length));
return new String(decryptedData, StandardCharsets.UTF_8);
}
```

-----Encryption calling method-----

```
String enKey = AppConfig.AES_key;
byte[] key = enKey.getBytes(StandardCharsets.UTF_8); // Convert key to byte array
//Encrypt JSON string

byte[] data = Aes256Encryption.encrypt(body.toString(), key);
//Encode encrypted data to Base64

String base64 = Base64.getEncoder().encodeToString(data);
```

-----Decryption calling method-----

```
byte[] key = enKey.getBytes(StandardCharsets.UTF_8);
//Decrypt the data

String data = Aes256Encryption.decrypt(Base64.getDecoder().decode(encryptedData), key);
```

-----Checksum-----

```
public static String
generateChecksum(String input) {try {
//Create a SHA-256 MessageDigest instance

MessageDigest digest = MessageDigest.getInstance("SHA-256");
//Convert input string to bytes

byte[] inputBytes = input.getBytes("UTF-8");
//Compute the hash

byte[] hashBytes = digest.digest(inputBytes);

//Convert hash bytes to hexadecimal string

StringBuilder hex = new
StringBuilder(hashBytes.length * 2);for (byte b
: hashBytes) {
hex.append(String.format("%02x", b));
}
return hex.toString();
}
catch (NoSuchAlgorithmException |
java.io.UnsupportedEncodingException e) {throw new
RuntimeException("Error computing hash", e);
}
}
```

===== END OF DOCUMENT =====