RAJ E-SIGN PROJECT,

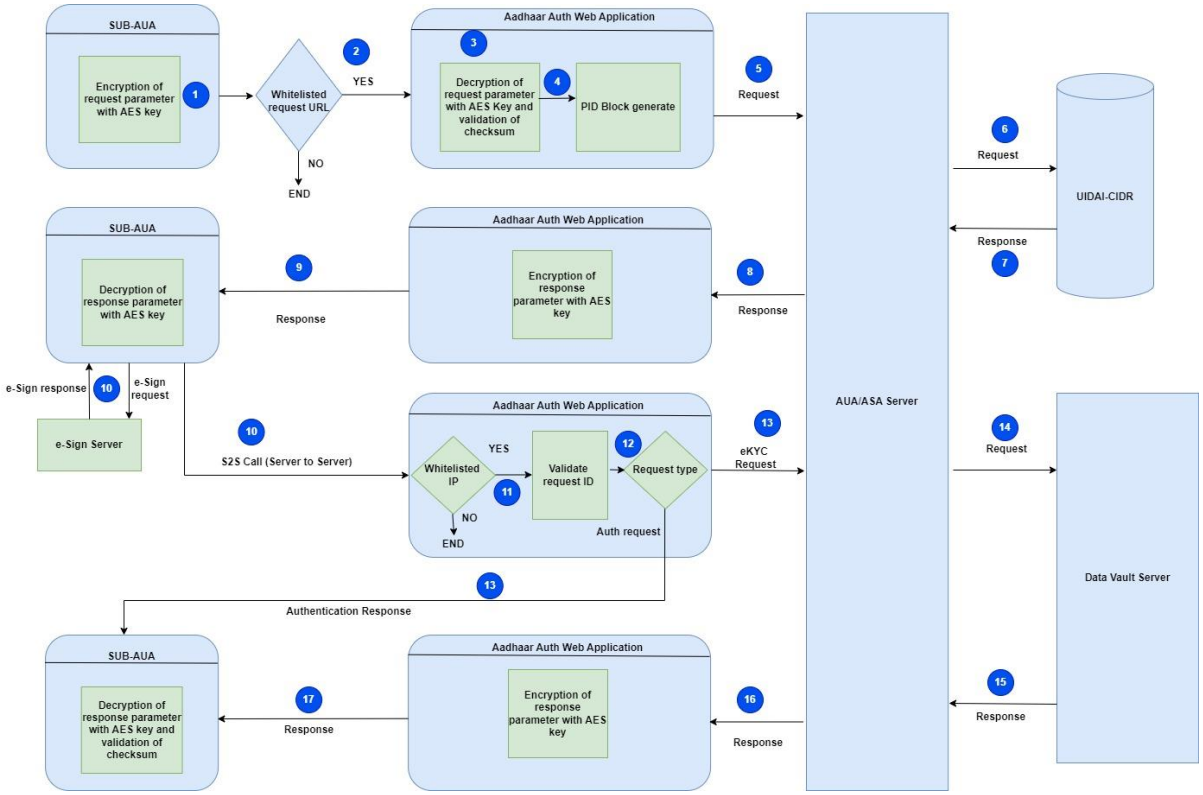
DEPARTMENT OF INFORMATION TECHNOLOGY & COMMUNICATION, GOR

e-Sign API 1.0 INTEGRATION DOCUMENT

(HTTP REDIRECTION MODE)

Introduction:- Presently, these Aadhaar authentication services are being provided through REST APIs. Now, AUA-DoIT&C is swiching 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



Transaction initiation

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 Data:**
- **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 >",
  "VerificationType": "<VerificationType >",
  "OperatorIdType": "<OperatorIdType >",
  "OperatorId": "<OperatorId >",
  "Purpose": "<Purpose >",
  }
- **Step 2:** Encrypt the JSON string created in step 1 using AES key.

<table>
<tr><th>SR. No.</th><th>Request Parameter</th><th>Parameter Type</th><th>Parameter Constraints</th><th>Description</th></tr>
<tr><td>1</td><td>SSOToken</td><td>string</td><td></td><td>SSO session Token for SSO session validation</td></tr>
<tr><td>2</td><td>AppCode</td><td>string</td><td>Required
Max Length: 32</td><td>Application-ID Issued by DoIT&C.</td></tr>
<tr><td>3</td><td>RequestId</td><td>String</td><td>Required
Length: 18</td><td>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: -
Sub-AUA Code + 8-digit Random number.
For Ex.- PRXXXXXXXX + 12345678</td></tr>
<tr><td>4</td><td>AadhaarNo</td><td>String</td><td>Required
Min Length: 12
Max Length: 15</td><td>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.</td></tr>
<tr><td>5</td><td>RequestType</td><td>Int</td><td>Required</td><td>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</td></tr>
<tr><td>6</td><td>ReturnURL</td><td>String</td><td>Required</td><td>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.</td></tr>
<tr><td>7</td><td>Checksum</td><td>Hash</td><td></td><td>fixed checksum hash of values AppCode, RequestId, AadhaarNo, RequestType</td></tr>
<tr><td>8</td><td>VerificationType</td><td>Int</td><td>Required for Face Authentication</td><td>Verification Type parameter defines user type who will perform Face Authentication. It accepts only two integer values as below: -
1 - Beneficiary
2 - Authentication Operator</td></tr>
<tr><td>9</td><td>OperatorIdType</td><td>Int</td><td>Required if Face Authentication is to be performed through an Authentication Operator</td><td>This defines the ID Type of Authentication operator who will perform Face Authentication in his mobile.
This field accepts only three integer values-
<table>
<tr><th>OperatorIdType</th><th>OperatorId</th></tr>
<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>
</table></td></tr>
</table>

<table>
<tr><td>10</td><td>OperatorId</td><td>String</td><td>Required if Face Authentication is to be performed through a Authentication Operator</td><td>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><tr><th>OperatorIdType</th><th>OperatorId</th></tr><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></table></td></tr>
<tr><td>11</td><td>Purpose</td><td>String</td><td>Required
Max length: 100</td><td>This parameter defines the purpose of the Aadhaar Authentication Authentication i.e. name of any schemes or any other purpose.</td></tr>
</table>

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

- 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=

PRODUCTION URL:

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=

PRODUCTION URL:

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=

PRODUCTION URL:

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/multipleESignedocument?client_id=

PRODUCTION URL:

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/pageWiseSignedocument?client_id=

PRODUCTION URL:

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=

PRODUCTION URL:

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=

PRODUCTION URL:

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	llx	String	X- coordinate of sign
8	lly	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)
{
EncriptionResult 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.Write(plainText);
}
model.IsSuccess = true;
model.EncriptedData = 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)
{
EncriptionResult 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.DecriptedData = 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[] EncriptedData { get; set; } = new byte[0];
public string DecriptedData { 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 ==========