Integration of eSign API Version 2.2 for ASP On-boarding

By RISL e-Governance Infrastructure Limited
(RISL e-Gov)

Version 2.2

Revision History

Revision Date	eSign Document ver	Description of Change(s)
01-June-2019	2.0	- Initial Draft of eSign 2.1 Integration.
15-January-2019	2.1	- Internally handling of intermediate file and tick image. - eSign on each page - eSign on specific page - Tick image visibility. - flexible stamping positions of signature
03-February-2020	2.2	-Multiple user e-signing functionality added

1 Contents

1 About RISL e-Gov

RajComp Info Services Ltd. (RISL), (formerly RajCOMP) is a Government of Rajasthan undertaking which was incorporated under Companies Act 1956 on 27-10-2010. Its primary areas of expertise include IT consultancy, e-Governance project conceptualization and implementation, capacity building in the IT area, provision of customized IT solutions spanning hardware as well as software projects, GIS development and multimedia development, RISL works under aegis of Principal Secretary, IT&C, Government of Rajasthan, who is the Chairman and Managing Director of the Company.

RISL is the State Designated Agency (SDA) for implementation of projects under National e-Governance Plan (NeGP) including i.e. State Data Centre (SDC), State Wide Area Network (SWAN), Common Service Center Delivery Gateway (SSDG) and other Mission Mode Projects (MMPs).

Some of the projects undertaken by RISL are -

a. State Data Center
b. State Wide Area Network
c. Common Service Center
d. e-District
e. e-Procurement
f. State Portal and State Service Delivery Gateway
g. Arogya Online
h. Rajasthan Public Service Commission Computerization
i. Online Recruitment Portal
j. Tourism Department
k. Scouts and Guide Department
l. Language and Library Department
m. Web based Lottery System
n. Rajasthan State Road Transport Corporation

1.1 Electronic Signature Service (eSign) Services

eSign facilitates electronically signing a document by a user using an Online service. Electronic Signature is created using authentication of customer through a national Unique Identification Number called Aadhaar. eSign has the potential to revolutionize the way business and governance is conducted and pave the way for a digital transformation into a paperless environment.Features of eSign Services

- 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

1.2 Benefits of eSign

- Saves cost and time
- User Convenience
- Legally recognized
- Suitable for individuals, businesses and Government
- Integrity with complete Audit trail

No need of physical dongle

1.3 Legality

Section 3 of The Information Technology (IT) Act, 2000 provides

- Legal sanctity to electronic signatures
- Electronic signatures are accepted at par with wet signatures

2 eSign API Specifications History

Below eSign version history released by CCA in India.

eSign API Version	Release date	Reference Links
2.1	04 May 2018	http://www.cca.gov.in/cca/sites/default/files/files/eSign-APIv2.1.pdf
1.6	22 April 2016	http://www.cca.gov.in/cca/sites/default/files/files/eSign-APIv1.6.pdf
1.5	17 Aug 2015	http://www.cca.gov.in/cca/sites/default/files/files/eSign-APIv1.5.pdf
1.0	15 July 2015	http://www.cca.gov.in/cca/sites/default/files/files/eSign-API%20v1.0.pdf

3 Integration of eSign API 2.1 Details

RISL e-Governance eSign service is exposed as stateless service over HTTPS. Usage of open data format in XML and widely used protocol such as HTTPS allows easy adoption and deployment of this service. To support strong end to end security and avoid request tampering and man-in-the-middle attacks, it is essential that the requests and responses are digitally signed.

The usage of HTTPS shall ensure transport layer encryption, while digital signing of XML shall ensure integrity & authenticity of data.

3.1 eSign Input Request XML

ASP has to submit the **eSign input xml request** to below RISL e-Gov URL

UAT URL	*http://59.145.20.75/esign/2.1/signdoc/*
Production URL	https://esign.rajasthan.gov.in/esign/2.1/signdoc/
Protocol	HTTPS
Method	POST
Content-Type / enctype	form-data (Multipart Form)
Input Parameters	"msg" – This parameter should contains the signed input xml request of the ASP which is as per the eSign 2.1 API specification.
Note	According to W3C XML specification there are some reserved entity ("amp", "lt", "gt", "apos", "quot") in xml. It should be in expanded form in value if any. References:- https://www.w3.org/TR/xmldsig-core/ https://www.w3.org/TR/xml11/#sec-predefined-ent

3.2 eSign Response XML

Below parameters will be received to the ASP application as eSign response of input xml request from RISL e-Gove eSign application.

Response URL	eSign response will be reverted to the **"responseUrl"** from the ASP input xml request.
Protocol	HTTPS
Method	POST
Content-Type / enctype	form-data
Input Parameters	"msg" – This parameter contains the document hash signature of the input xml request i.e. eSign response.

3.3 Pre-requisite of ASP .Net utility

1. Visual Studio 2017
2. .Net version 4.6
3. Digital Signature Certificate (V3 version) (No self-signed in production environment)
    a. Valid ASP ID should be created at ESP end before go live.
    b. Should be a valid certificate issued by a CA licensed under Information technology (IT) Act.
    c. Should be either an Organizational Person Digital Signature Certificate or an Organizational Document Signer Certificate. The O value in the certificate should be the legal entity name of the ASP organization.
    d. Should be either Class 2 or Class 3 certificate.
    e. Should be valid for at least six months from date of submission
    f. ASP can procure DSC from any of Certifying Authority (CA) as listed on the CCA website as mentioned below at http://www.cca.gov.in

4 RISL Application Technical Documentation for DLL file implementation

- RISL has built a .NET utility for smooth integration of all its on boarding ASPs for eSign 2.1 API integration as per CCA guidelines. Please refer below implementation details. After extracting downloaded ZIP, ASP will find below details:
    - Stepwise integration document
    - DLL
    - Sample demo pages for easier integration.
- DLL and Sample demo pages contains below list of functionalities
    - Request creation as per eSign 2.1 API specification
        - For input pdf document (Fully qualified path)
        - For input document hash (e.g PDF)
    - Signing the request xml using Certificate i.e. *.pfx
    - Stamping the signature response for PDF document.

- This utility will read document e.g. pdf file which is to be signed from specified fully qualified path or hash of the document which needs to be signed. After reading document i.e. pdf file we are inserting blank signature box in pdf. This step will not be applicable if document is other than Pdf. Next step to create hash of that pdf in which we have added blank signature (if applicable).

- ❖ Prepare eSign request xml and signed it with ASP's Certificate [.pfx] as per the ASP application.
- ❖ Further submit the generated request to above UAT or production environment for generation of signature at eSign server.

- ❖ Below list of functions available in the Dll utility. ASP can utilize any of them as per their business needs.

Sr. No.	Method Name	Method Attributes	Method Description	Document Type
1.	signDetachedStage1	arg1: DocBase64 (string)	Generate Document Hash with the signature appearance.	PDF
2.	signDetachedStage2	arg1: signature (string)	Stamp signature response (success) received from ESP in the PDF document only.	PDF
3.	SignXml	arg1: xml (XmlDocument) arg2: certificate (X509Certificate2)	Generate un-signed request xml. Signed generated request xml using ASP's Certificate i.e. pfx	PDF
4.	visibleSign	arg1: certificate (string) arg2: lowerx_coordinate (int) arg3: lowery_coordinate (int) arg4: pageNotobeSigned (int) arg5: signAllPages (bool)	Make the signature visible.	NA

Note: Kindly refer complete published method summary below in this document.

4 Method Accessing Details

5.1 Method calling Details

- ❖ For calling specific method kindly pass all the required parameters as mentioned below.
- ❖ Exception Handling – ASP needs to handle the exception at their application end.
- ❖ Mapping details of input parameter and generated request xml using JAR for reference.

Sr. No.	Input parameter to JAR	Mapping with Esign 2.1 request xml
1.	aspId	aspId

2.	responseUrl	responseUrl
3.	ekycId	ekycId
4.	txn	Txn
5.	authMode	authMode

5.2 Published Method Summary Details

5.2.1 `signDetachedStage1` Method Details

Method Description: - Generate Document Hash with the signature appearance.

Method Return Type Signed input request xml in string format using ASP's certificate.

Exception Handling User Need to be handle the exception at their end.

Method Prototype

```
signDetachedStage1(string base64InFile);
```

Mandatory Input Parameter(s)

Sr. No.	Item	Description
1.	`base64InFile`	Base64 of fully qualified input pdf

5.2.2 Creation of Input xml:

For generating signed input request xml using ASP's certificate i.e. pfx. This method is applicable only for PDF document Type. This method accepts fully qualified input pdf path for which signature needs to be generated. By default, ASP application will receive signature response with pkcs7 signature type after successful eSign, if request is generated using this method.

Sr. No.	Item	Description
1.	eSign Request format	<Esign ver="" sc="" ts="" txn="" ekycId="" ekycIdType="" aspId="" AuthMode="" responseSigType="" responseUrl=""> <Docs> <InputHash id="" hashAlgorithm="" docInfo="">Document Hash in Hex</InputHash> </Docs> <Signature>Digital signature of ASP</Signature> </Esign>
2.	Ver	2.1
3.	asp_id	Asp id of organization provided by RISL.
4.	Sc (ASP should taken a clear consent from 'eSign user' to carry on eSign from their front ending application.)	This data should be used to fill the "sc" in the request. Only valid value is "Y".
5.	Txn	ASP specific transaction identifier.

		This is similar to “txn” value in eKYC request but there is no validation at eSign application with respect to the format. Only duplicate txn ids would be rejected.
6.	ekycId (. ASP may collect UIDToken from the end user. This is optional.)	If the ASP collects UIDToken, this should be passed in the parameter "eKYCId". Otherwise, eKYCID should be ""
7.	authMode	1” for OTP,“2” Biometric and “3” for Iris
8.	resp_url	ASP Active URL to receive esign response xml,
9.	respSignatureType	ASP required RISL response signature type as per CCA mentioned signature type. **Note:** If you do not want to provide response signature type then pass it as a blank or null, internally by default you will get pkcs7 signature type. In case ASP pass incorrect response signature type then by default it will be pkcs7. Allowed values are “rawrsa”, “pkcs7” and “rawecdsa”.

This data is used to fill the details of the <Docs> tag.

Sr. No.	Item	Description
1.	Id	Value should be 1. Only single document hash is allowed
2.	HashAlgorithm	Should be fixed to “SHA256”
3.	DocInfo	Description for the respective document being signed, not more than 50 characters.
4.	InputHash	Compute the SHA256 hash value of the document submitted by the end user in Hex format

5.2.3 `SignXml` Method Details

This method will sign the generated request xml (outside this utility) using ASP’s private certificate i.e. pfx

This method is applicable only for PDF document Type.

Method Return Type Signed esign request xml in string format.

Exception Handling User Need to be handle the exception at their end.

Method Prototype

```
SignXml(XmlDocument xmlDoc, X509Certificate2 uidCert)
```

Mandatory Input Parameter(s)

Sr. No.	Item	Description
1.	`xmlDoc`	Generated Input xml
2.	`uidCert`	ASP certificate (PFX certificate)

5.2.4: `signDetachedStage2` Method Details: -

Digesting eSign response will vary from application to application and the respective ASP has to handle this part as per their application structure. However, below sample has been provided to collect response data from ESP API which will pass response to method "**`signDetachedStage2`**". Method will affix signature in PDF at pre-define co-ordinate. Below sample code for reference.

This method will stamp signature response (success) received at ASP.

This method is applicable only for PDF document Type.

Method Return Type Signature insertion message in string format in case of success.

Exception Handling User Need to be handle the exception at their end.

Method Prototype

```
signDetachedStage2(string finalSign)
```

Mandatory Request Parameters: -

Sr. No.	Item	Description
1.	`finalSign`	Base64 Signature byte

5.2.4 `visibleSign` Method Details:- This the final method for making the signature visible.

Exception Handling User Need to be handle the exception at their end.

Method Prototype

`visibleSign(string UserX509Certificate, int lowerx_coordinate, int lowery_coordinate, int pageNo, bool signallPages)`

Mandatory Request Parameters: -

Sr. No.	Method Attribute	Description
1.	`UserX509Certificate`	Certificate receive in the response
2.	`lowerx_coordinate`	Lower X coordinate of PDF to stamp the visible signature
3.	`lowery_coordinate`	Lower Y coordinate of PDF to stamp the visible signature
4.	`pageNo`	Page Number of the PDF on which the signature should be printed (will work only when signAllPages attribute is set as false)
5.	`signAllPages`	To print the signature on all pages, set this attribute to true else false

5 Annexure I – eSign Service Error Codes

Below list of error codes can be occurred during processing of esign request.

S.No.	Code No	Description
1.	ESP-001	Internal Error ESP Service Unavailable
2.	ESP-002	Internal Error ESP Service Unavailable
3.	ESP-408	Invalid ASP Request!
4.	ESP-410	Invalid ASP Consent or Consent Not provided!
5.	ESP-415	Invalid OTP. No Of Retry Attempt Exhausted!!
6.	ESP-416	Invalid Biometric. No Of Retry Attempt Exhausted!!
7.	ESP-417	Invalid Iris.
8.	ESP-901	Invalid Authentication Mode
9.	ESP-902	Invalid ASP ID. It cannot be Empty
10.	ESP-903	Invalid ASP ID. It may not exist or may be inactive.
11.	ESP-905	Document Hash not received
12.	ESP-906	UID Token does not match
13.	ESP-907	Request Time Stamp cannot be Empty
14.	ESP-908	Request Time Stamp is not valid. Please check the server time.
15.	ESP-909	Transaction ID cannot be Empty
16.	ESP-910	Duplicate Transaction ID for the given ASP.
17.	ESP-911	Input XML Signature verification failed.
18.	ESP-944	User terminated eKYC process
19.	ESP-945	User terminated eKYC process after OTP Generation
20.	ESP-946	User interface page expired
21.	ESP-922	Invalid Signature on Input XML. Please use the corresponding certificate mapped with ESP
22.	ESP-991	ESP Database Connectivity Error
23.	ESP-992	Input XML Parsing Error.
24.	ESP-993	Error Parsing CA Response XML
25.	ESP-999	Unknown Error. Esign failed.
26.	ESP-000	Malformed/Not-Well Formed/Malicious XML Request or xml request size exceeds 20 KB

In addition to above error codes, please find list of e-KYC error codes in the provided link, which may occur during the esign processing.

URL: - Refer error section from document from the links mentioned below

Sr. No	API Name	URL of API Documentation
1	OTP API Ver - 2.5	https://uidai.gov.in/images/resource/aadhaar_otp_request_api_2_5.pdf
2	eKYC API Ver - 2.5	https://uidai.gov.in/images/resource/aadhaar_ekyc_api_2_5.pdf
3	Auth XML Ver – 2.5	https://uidai.gov.in/images/resource/aadhaar_authentication_api_2_5.pdf

6 Assumptions/ Limitations

- Please note that ASP has to take a complete responsibility of error handling in their application. Also, if esign process failed, then application behavior and error message to be displayed to end user should be with ASP end only. E-authentication server will only send valid success or failure response along with error codes (if any) to ASP applications.

- Shared sample code/utility is only for reference to ASP. ASPS have to implement required details by themselves as per their application requirement and architecture of that application.

- Any changes in ESP application based on real time scenarios including error conditions will be validated time to time as per the application demand and same will be communicated to ASP if required in future. ASP has to handle/include such scenarios in their application if not handled.

- All scenarios including success and failure handling should be tested properly by integrating entity in their own UAT environment before go live.

<End of Document>