Standard interface

Introduction

The REST interface offered by SWS is exposed at the path:

```
http://<IP-APPLIANCE>:8080/SignEngineWeb/rest
```

This path is composed by other sub-path for every of purpose:

- **admin**: method for sws like remove certificate from cache
- **enquiry**: contain the method for obtain the information like signatures available, signer certificate, timestamps available ecc...
- **sign**: this is the principal path of SWS and contain the methods for apply the signature
- **timestamps**: methods for apply the timestamp on every type of file

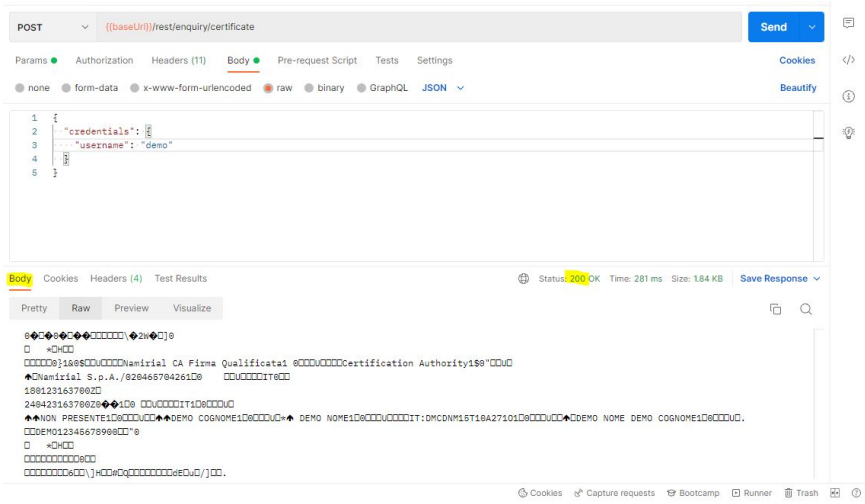
And in this guide will be described how manage the error generated by the REST interface.

NOTE: this interface is available from SWS version: 2.5.52

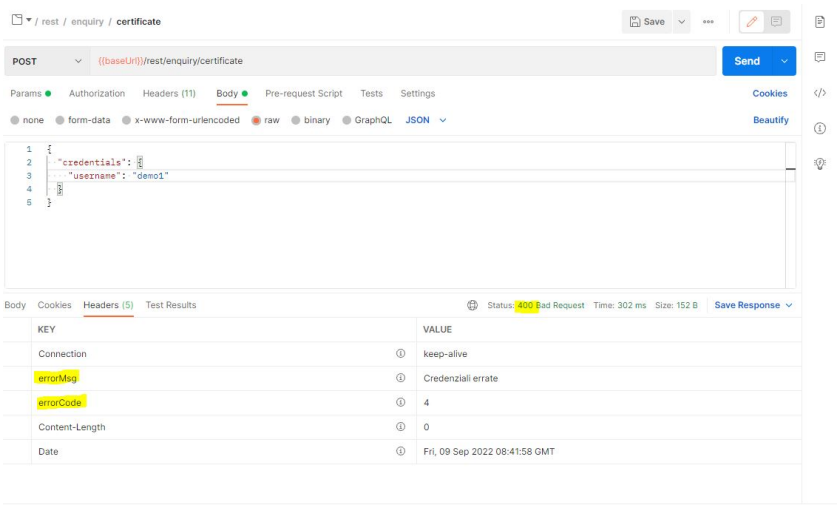
Convention (manage the response)

SWS rest interface use this convention for create the response

Request is CORRECT, will return response code 200 with response body (if present) . Like in this example:



Request NOT-CORRECT with error managed, will return response code 400 and the header will have the field "errorMsg" with error description (in Italian) and field "errorCode" with code error. Like in this example:



NOTE: if you want the "errorMsg" in a specified language, you can use the method "enquiry/errors" will be described in the next section.

Enquiry

ENQUIRY: certificate

Description	return the certifcate associated to "device_signer"
HttpMethod	**POST**
Path	`/enquiry/certificate`

Request	{
	"credentials": {
	"username": "device_signer"
	}
	}

Response	return the byte array of certificate associated to device_signer

ENQUIRY: signatures

Description	return the numer of signatures apposed from "device_signer"	
HttpMethod	**POST**	
Path	`/enquiry/signatures`	

Request	{
	"credentials": {
	"username": "device_signer"
	}
	}	

Response	Number of signatures apposed	

ENQUIRY: signatures-available

Description	return the number of signatures which "device_signer" can apply
HttpMethod	**POST**
Path	`/enquiry/signatures-available`

Request	{
	"credentials": {
	"username": "device_signer"
	}
	}

Response	Number of signatures available

ENQUIRY: otps

Description	return the otp list associated to "device_signer"
HttpMethod	**POST**
Path	`/enquiry/otps`

Request	{
	"credentials": {
	"username": "device_signer"
	}
	}

Response	[
	{
	"idOtp": number,
	"serialNumber": "string",
	"type": "otp-type-enum"
	},
	{
	"idOtp": number,
	"serialNumber": "string",
	"type": "otp-type-enum"
	}
	]

ENQUIRY: timestamps-available

Description	return the numeber of timestamp available (valid only for Namirial TSA account)
HttpMethod	**POST**
Path	`/enquiry/timestamps-available`

Request	{
	"timestampUrl": "timestamp-namirial-enquiry-url",
	"timestampUsername": "tsa-username",
	"timestampPassword": "tsa-password"
	}

Response	Number of timestamps available

ENQUIRY: errors

Description	return the error description associated to error code
HttpMethod	**POST**
Path	`/enquiry/errors`

Request	{
	"error_code": integer,
	"lang": "COUNTRY-CODE-2DIGIT"
	}

Response	[
	{
	"errorCode": integer,
	"errorLanguage": "CONUNTRY-CODE-2DIGIT",
	"errorLanguage2": "COUNTRY-CODE-3DIGIT",
	"errorText": "Description error in language"
	}
	]

ENQUIRY: all-signature-fields-with-preferences

Description	return a list of SignatureFieldName
HttpMethod	**POST**
Path	`/enquiry/all-signature-fields-with-preferences`
Header	Content-Type = multipart/form-data Accept = application/json
Request	

preferences	<pre>{
	    "withDetails": boolean,
	    "withCertificate": boolean,
	    "encryptionPassword": string
	}</pre>

buffer	PDF file to extract field

Response	<pre>[
	    {
	        "identifier": string,
	        "signatureDetails": {
	            "name": string,
	            "signDate": unixtime,
	            "location": string,
	            "reason": string,
	            "page": integer,
	            "appearance": {
	                "width": float,
	                "height": float,
	                "x": float,
	                "y": float
	            },
	            "certificate": "<base64-encoded certificate>",
	            "subjectDN": "string"
	        },
	        "signed": boolean
	    },
	        ...
	]</pre>

ENQUIRY: available-signature-fields

Description	return a list with name of signature field
HttpMethod	**POST**
Path	`/enquiry/available-signature-fields`
Header	Content-Type = multipart/form-data Accept = application/json
Request	
buffer	PDF file to extract field
encryptionPassword	password to open PDF if present

Response	<pre>[
	    "FieldName-1",
	    "FieldName-2",
	        ...
	]</pre>

Admin

ADMIN: remove-certificate-from-cache

Description	remove the certificate from cache of SWS
HttpMethod	**PUT**
Path	`/admin/remove-certificate-from-cache`

Request	{
	"error_code": integer,
	"lang": "COUNTRY-CODE-2DIGIT"
	}

Response	

Timestamps

TIMESTAMPS: apply

Description	permits to apply timestamp on specified file
HttpMethod	**POST**
Path	`/timestamps/apply`
Request	

timestampPreferences	{
	"filenameInTSD": "string",
	"outputAsPDF": boolean,
	"outputAsTSD": boolean,
	"outputBase64Encoded": boolean,
	"timestampHashAlgo": "string",
	"timestampPassword": "string",
	"timestampUrl": "string",
	"timestampUsername": "string"
	}

content	file to apply timestamp
Response	

User

USER: change-password

Description	permits to change the password associated to device signer
HttpMethod	**POST**
Path	`/user/change-password`

Request (for remote device signer)	{
	"credentials": {
	"idOtp": idOtp or -1,
	"otp": "otpCode",
	"password": "old-password-of-device-signer-remote",
	"username": "device-signer-remote"
	},
	"newPassword": "new-password-of-device-signer-remote"
	}

Request (for automatic device signer)	{
	"credentials": {
	"securityCode": "securityCode associate to automatic device signer",
	"password": "old-password-of-device-signer-automatic",
	"username": "device-signer-automatic"
	},
	"newPassword": "new-password-of-device-signer-automatic"
	}

Response	Password update succesfully

Sign

SIGN: openSession

Description	permits to open the sessione for apply multiple sign with remote device
HttpMethod	**POST**
Path	`/sign/openSession`

Request	<pre>{
	  "credentials": {
	  "idOtp": -1,
	  "otp": "775351",
	  "password": "12345678",
	  "username": "RHIP22021116852552"
	  }
	}</pre>

Response	String with the session

SIGN: getRemainingTimeForSession

Description	permits to obtain the time until the session is valid
HttpMethod	**POST**
Path	`/sign/getRemainingTimeForSession`

Request	<pre>{
	  "credentials": {
	  "sessionKey": "zZto1G0DpL
	/vBFkTnK7caquzY5pasOlzS+bQG7wUkOONnbV7Vhd+JSPTjP7ZqTYR12QjS0W89T7UmnQB2KzAQ3C4NalDgFE67ntqoGm7uO
	U7+oOPLvKQv/p5aeZ2bcjKe6x5KQPUEH//rKaExFcLcLj8cnwXfFBixJ4MN+3o8S5535HcRxWv+YoTHHgAY16Fh0yJGfLL3x
	/4W+HJeiIYL2cHpKNTGkKcGTM8Eon0R+djNFvKzZSF1VIETPADqDdvgLYkRWODd3yoUvExGk5BcQKVm0Z7Nd7NMKl4NRbHu
	mdqmqy81jchQv2qlXIxSpjZ0GTnL4vDZMF2MP2DGHPoWw==",
	  "username": "RHIP22021116852552"
	  }
	}</pre>

Response	Seconds until the session is valid

SIGN: closeSession

Description	permits to destroy the session before will expire
HttpMethod	**POST**
Path	`/sign/closeSession`

Request	<pre>{
	  "credentials": {
	  "sessionKey": "zZto1G0DpL
	/vBFkTnK7caquzY5pasOlzS+bQG7wUkOONnbV7Vhd+JSPTjP7ZqTYR12QjS0W89T7UmnQB2KzAQ3C4NalDgFE67ntqoGm7uO
	U7+oOPLvKQv/p5aeZ2bcjKe6x5KQPUEH//rKaExFcLcLj8cnwXfFBixJ4MN+3o8S5535HcRxWv+YoTHHgAY16Fh0yJGfLL3x
	/4W+HJeiIYL2cHpKNTGkKcGTM8Eon0R+djNFvKzZSF1VIETPADqDdvgLYkRWODd3yoUvExGk5BcQKVm0Z7Nd7NMKl4NRbHu
	mdqmqy81jchQv2qlXIxSpjZ0GTnL4vDZMF2MP2DGHPoWw==",
	  "username": "RHIP22021116852552"
	  }
	}</pre>

Response	

SIGN: sendOtpBySMS

Description	permits to destroy the session before will expire
HttpMethod	**POST**
Path	`/sign/sendOtpBySMS`

Request	{
	"credentials": {
	"username": "RHIP22021116852552"
	}
	}

Response	

SIGN: signCades

Description	permits to apply the cades signature
HttpMethod	**POST**
Path	`/sign/signCades`
Request	

credentials	{
	"username":"device signer name",
	"password":"PIN of device signer name",
	"idOtp":associated to device signer or -1,
	"otp":"otp code",
	"sessionKey":"string with sessionKey"
	}

cadesPreferences	{
	"filenameInTSD": "string",
	"outputAsPDF": boolean,
	"outputAsTSD": boolean,
	"outputBase64Encoded": boolean,
	"timestampHashAlgo": "string",
	"timestampPassword": "string",
	"timestampUrl": "string",
	"timestampUsername": "string",
	"hashAlgorithm": "string",
	"level": "enum",
	"withTimestamp": boolean,
	"counterSignature": true,
	"counterSignatureIndex": 0,
	"detached": boolean
	}

buffer	file to sign
Response	byte array of signed files

SIGN: signCades (detached output p7s)

If you want make the Cades detached signature, SWS not require all files to sign, but only the hash. The tag "buffer" will be the hash of the file.

For example if we want the cades detached signature of this PDF the procedure is:

Calculate the hash of this file, for example with the openssl:

```
openssl dgst -sha256 -binary FILE_TO_BE_SIGN | openssl enc -a
```

And in output will obtain the hash to sign, will be:

```
HASH TO SIGN = msj3f4hJCSELbMkWjkFwNrf0XhkebTnAKaKhx4686DY=
```

Now you can decode this string and will be the input relates to field "buffer"

This string "msj.....DY=" decoded will be the "buffer" on REST signCades like this file (this it the byte array to sign)

Description	permits to obtain the cades detached signature (p7s) , from the hash associated to the file to sign
HttpMethod	**POST**
Path	`/sign/signCades`
Request	

credentials	{
	"username":"device signer name",
	"password":"PIN of device signer name",
	"idOtp":associated to device signer or -1,
	"otp":"otp code",
	"sessionKey":"string with sessionKey"
	}

cadesPreferences	{"detached": true}
buffer	btye array relates to the hash files to sign
Response	byte array related to sign of the hash and the certificate associate

REST RESPONSE:

In output will obtain the hash signed and the certificate associated to the private key which has signed the hash, the response will be this

SIGN: signPades

Description	permits to apply the pades signature
HttpMethod	**POST**
Path	`/sign/signPades`
Request	

credentials	{
	"username":"device signer name",
	"password":"PIN of device signer name",
	"idOtp":associated to device signer or -1,
	"otp":"otp code",
	"sessionKey":"string with sessionKey"
	}

padesPreferences	{
	"filenameInTSD": "string",
	"outputAsPDF": boolean,
	"outputAsTSD": boolean,
	"outputBase64Encoded": boolean,
	"timestampHashAlgo": "string",
	"timestampPassword": "string",
	"timestampUrl": "string",
	"timestampUsername": "string",
	"hashAlgorithm": "string",
	"level": "enum",
	"withTimestamp": boolean,
	"encryptInAnyCase": boolean,
	"encryptionPassword": "string",
	"lockFields": [
	"string"
	],
	"needAppearanceDisabled": boolean,
	"page": 0,
	"signerImage": {
	"fieldName": "string",
	"fontName": "string",
	"fontSize": 0,
	"image": "string",
	"imageFilename": "string",
	"imageURL": "string",
	"imageVisible": boolean,
	"location": "string",
	"reason": "string",
	"scaled": true,
	"signerName": "string",
	"textPosition": "enum",
	"textVisible": boolean,
	
	"scaledText": boolean,
	"width": int,
	
	"height":int,
	"x": int,
	"y": int
	},
	"signerImageReference": "string",
	"withSignatureField": boolean
	}

image	file with image (of appereance)
buffer	PDF file to sign
Response	byte array of signed files

SIGN: signPadesMultiFieldName

Description	permits to apply the pades signature ONLY on PDF with signatures fields already exist
HttpMethod	**POST**
Path	`/sign/signPadesMultiFieldName`
Request	

credentials	{
	"username":"device signer name",
	"password":"PIN of device signer name",
	"sessionKey":"string with sessionKey"
	}

padesPreferences	{
	
	"withSignatureField": true
	"outputBase64Encoded": boolean,
	"timestampHashAlgo": "string",
	
	"fieldsNameList": list_of_signatures_fields (ex, ["Signature-Field-1", "Signature-Field-2"],
	
	"signAllFields": boolean,
	
	"timestampPassword": "string",
	"timestampUrl": "string",
	"timestampUsername": "string",
	"hashAlgorithm": "string",
	"level": "enum",
	"withTimestamp": boolean,
	"encryptionPassword": "string",
	"signerImage": {
	
	"fieldName": "string",
	"fontName": "string",
	"fontSize": 0,
	"image": "string",
	"imageFilename": "string",
	"imageURL": "string",
	"imageVisible": boolean,
	"location": "string",
	"reason": "string",
	"scaled": boolean,
	"signerName": "string",
	"textPosition": "enum",
	"textVisible": boolean,
	
	"scaledText": boolean,
	
	},
	}

image	file with image (of appereance)
buffer	PDF file to sign
Response	The body contain the byte array of files signed fully or partially
Response code	200: the file is signed fully 400: the request isn't correct. The header params: "errorMsg" and "errorCode" contains the details about the errors 422: the file is signed partially and the header params "remainingFieldNames" contains the list of unsigned param. The param "errorCode" and "errorMsg" contain details about the error 500: an internal server error has occured.

SIGN: signXades

Description	permits to apply the xades signature
HttpMethod	**POST**
Path	`/sign/signXades`
Request	

credentials	{
	
	"username":"device signer name",
	
	"password":"PIN of device signer name",
	
	"idOtp":associated to device signer or -1,
	
	"otp":"otp code",
	
	"sessionKey":"string with sessionKey"
	
	}

xadesPreferences	{
	"filenameInTSD": "string",
	"outputAsPDF": boolean,
	"outputAsTSD": boolean,
	"outputBase64Encoded": boolean,
	"timestampHashAlgo": "string",
	"timestampPassword": "string",
	"timestampUrl": "string",
	"timestampUsername": "string",
	"hashAlgorithm": "string",
	"level": "enum",
	"withTimestamp": boolean,
	"detached": boolean,
	"detachedReferenceURI": "string",
	"signElement": "string",
	"signatureId": "string",
	"withoutSignatureExclusion": boolean,
	"xPathQuery": "string"
	}

buffer	XML file to sign
Response	byte array of signed files

SIGN: signPKCS1

Description	permits to apply the raw signature (PKCS1)
HttpMethod	**POST**
Path	`/sign/signPKCS1`
Request	

credentials	{
	"username":"device signer name",
	"password":"PIN of device signer name",
	"idOtp":associated to device signer or -1,
	"otp":"otp code",
	"sessionKey":"string with sessionKey"
	}

signPreferences	{
	"hashAlgorithm": "enum"
	}

buffer	hash to sign
Response	byte array associated to hash signed

Verify

VERIFY: signatures

Description	permits to verify the signatures
HttpMethod	**POST**
Path	`/verify/signatures`
Request	
signedContent	file to verify

preferences	{
	"detachedContent": "string",
	"language": "COUNTRY_CODE_2_DIGIT" (es: IT),
	"pdfEncryptionPassword": "string",
	"recursive": true,
	"verifyOnDate": "YYYY-mm-dd" (for example: 2022-10-24)
	}

Response	Report with verify, this is a complex object: "SignedDocumentReportBean"

Verify timestamp

With SWS is possible to verify TSD and TSR using the preferences, below will be described the REST request.

VERIFY: tsr or tsd

Description	permits to verify the timestamps in tsd or tsr format
HttpMethod	**POST**
Path	`/verify/timestamps`
Request	
timestampedContent	file with timestamp
detachedContent	file original, where timestamp has ben applied (use this field only if you are verifying TSR)

preferences	{
	"responseWithoutContent": boolean,
	"language": "COUNTRY_CODE_2_DIGIT" (es: IT)
	}

Response	Return a complex object "TimestampReportBeanSummary"

VERIFY: tsd

Description	permits to verify the timestamps
HttpMethod	**POST**
Path	`/verify/timestamps/tsd`
Request	
tsd	timestamp to verify

preferences	{
	"responseWithoutContent": boolean,
	"language": "COUNTRY_CODE_2_DIGIT" (es: IT)
	}

Response	Return a list of complex objects: "TimestampReportBean"

VERIFY: tsr

Description	permits to verify the timestamps
HttpMethod	**POST**
Path	`/verify/timestamps/tsr`
Request	
tsr	timestamp to verify
content	file original, where timestamp has ben applied

preferences	|
	
	{
	"responseWithoutContent": boolean,
	"language": "COUNTRY_CODE_2_DIGIT" (es: IT)
	}

Response	Return a complex object "TimestampReportBean"