A-Trust Gesellschaft für Sicherheitssysteme
im elektronischen Datenverkehr GmbH
Landstraßer Hauptstraße 1b
The Mall E02
A-1030 Wien

https://www.a-trust.at
E-Mail: office@a-trust.at

Tel: +43 (1) 713 21 51 - 0
Fax: +43 (1) 713 21 51 - 350

User manual
a.sign premium seal qualified

(english translation)

Version: 1.2
Date: 07.04.2023

Contents

Date	Vers.	Author	Changelog
07.04.2023	1.2	Patrick Hagelkruys	textual changes
06.04.2023	1.1	Patrick Hagelkruys	add batch signature, see chapter 3.4
06.04.2023	1.0	Patrick Hagelkruys	new A-Trust design
24.05.2019	0.4	Patrick Hagelkruys Ramin Sabet	textual changes
24.05.2019	0.3	Patrick Hagelkruys	add errorocdes
23.05.2019	0.2	Patrick Hagelkruys	add live and test system urls add link to github add detail description of signature format
23.05.2019	0.1	Patrick Hagelkruys	first version

Table 1: document history

1 Overview

1.1 Summary

`a.sign premium seal qualified` is a qualified electonic seal in accordance with the eIDAS regulation [Cou14, Section 5, Article 38, Qualified certificates for electronic seals].

`a.sign premium seal qualified` is a remote signature, therefore the seal private key is stored in a hardware security module in the data center of A-Trust. For each signature the client software has to issue a request with the data to be signed to A-Trust. This request has to be signed with an authentication key.

To order an `a.sign premium seal qualified` you will need to generate a PKCS#10 request for an authentication certificate, that will later authenticate every signing request to the A-Trust servers.

This document describes the registration process and signature interface for the `a.sign premium seal qualified`.

2 a.sign premium seal qualified registration process

2.1 Preconditions

For `a.sign premium seal qualified` a PKCS#10 request for an authentication certificate is required.

This request can be generated using openssl using the following commands: a new private key file `private_key.pem` and a certificate request file `certificate_request.txt`.

```
openssl req -nodes
            -new
            -newkey rsa:2048
            -sha256
            -out certificate_request.txt
            -keyout private_key.pem
```

will be generated. Pay attention to the file `private_key.pem` file, it will be needed for every signature request and should be kept secret.

2.2 Order process

To order an a.sign premium seal qualified certificate visit the website https://www.a-trust.at/Bestellungen/asignsealqualified/, fill out the form and provide the necessary documents. An A-Trust Support employee will contact you regarding further action to complete the registration process.

After completing the registration process, A-Trust will provide you with an `a.sign premium seal qualified` certificate and an authentication certificate.

2.3 Optional step, combining private key and authentication certificate

If your client software needs a PKCS#12 file rather than a seperate private key and certificate file, the following command creates the PKCS#12 file with the name `authentication_certificate.p12`

```
openssl x509 -inform der
             -in SealAuthenticationCertificate.cer
             -out SealAuthenticationCertificate.pem

openssl pkcs12 -export
               -out authentication_certificate.p12
               -inkey private_key.pem
               -in SealAuthenticationCertificate.pem
```

3 a.sign premium seal qualified signature interface

3.1 Overview

To communicate with the A-Trust server a REST interface (HTTP POST and HTTP GET) is used.

3.2 Get certificate information

This call downloads the seal certificate for the specified serial number.

URL schema

```
/SealQualified/v1/Certificate/[authcertserial]/[sessionid]
```

Listing 1: get certificate uri schema

The parameter `authcertserial` is the certificate serial number of the authentication certificate in decimal format. The parameter `sessionid` is assigned by the calling application and is used to connect the sessions between client and server.

Request

```
GET /SealQualified/v1/Certificate/1058733338/sessionid
Host: ...
```

Listing 2: get certificate request

Response

```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache
Content-Disposition: attachment;filename=asignSealQualified.cer
Content-Length: 1077
Content-Type: application/x-x509-ca-cert

[binary data]
```

Listing 3: get certificate response

3.3 Sign hash value

This call performs a signature with the seal private key.

The data to be signed has to be hashed by the client before transmission to the A-Trust server.

See chapter 4.2 for the ECDSA signature format.

URL schema

```
/SealQualified/v1/Sign/[sessionid]
```

Listing 4: sign uri schema

The parameter `sessionid` is assigned by the calling application and is used to connect the sessions between client and server.

Request

```
POST /SealQualified/v1/Sign/sessionid HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 952

{
  "AuthSerial": "1058733338",
  "AuthCert": "MIIEiTCCA3GgAwIBAgI....",
  "Hash": "Dza/4+d+pQzueNkiecsIU+qXv...",
  "HashSignatureMechanism": "SHA256withRSA",
  "HashSignature": "wHP+yGEq8g9GT/IE..."
}
```

Listing 5: sign hash value request

The parameter `AuthSerial` contains the certificate serial number of the authentication certificate in decimal format. Instead of the `AuthSerial` you can also add the complete authentication certificate in base64 format as `AuthCert`.

`Hash` contains the to-be-signed hash value and `HashSignature` contains the signature with the authentication certificate over the hash value. `HashSignatureMechanism` specifies the algorithm used to sign the hash value.

For possible values for `HashSignatureMechanism` see chapter 4.1.

Response

```
HTTP/1.1 200 OK
Content-Length: 130
Content-Type: application/json; charset=utf-8

{
  "Signature":"BC4jJ\/fdAvBBln+y6h...egC7U="
}
```

Listing 6: sign hash value response

3.4 Sign a list of hash values

This call performs a signature with the seal private key for a list of hash values.

The data to be signed has to be hashed by the client before transmission to the A-Trust server.

See chapter 4.2 for the ECDSA signature format.

URL schema

```
/SealQualified/v1/Batch/Sign/[sessionid]
```

Listing 7: batch sign uri schema

The parameter `sessionid` is assigned by the calling application and is used to connect the sessions between client and server.

Request

```
POST /SealQualified/v1/Sign/sessionid HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 6572

{
  "AuthCert": null,
  "AuthSerial": "327876953",
  "ListOfHashes": "W3siSWQiOjAsIkhhc2giOiJrd2....0az0ifV0=",
  "HashSignature": "LMpY5AvMj5ZH...hcw==",
  "HashSignatureMechanism": "SHA256withRSA"
}
```

Listing 8: batch sign hash value requst

ListOfHashes

```
[
  { "Id": 0, "Hash": "rUnyi2UlFjL...8sjsE=" },
  { "Id": 1, "Hash": "HiW1B4....ecluI5Xw=" },
  { "Id": 2, "Hash": "Q2kZ2k9...Y7iiw=" },
  ...
  { "Id": 99, "Hash": "/m9UI6...ywN0eTmRVlg=" }
]
```

Listing 9: batch sign list of hashes

The parameter `AuthSerial` contains the certificate serial number of the authentication certificate in decimal format. Instead of the `AuthSerial` you can also add the complete authentication certificate in base64 format as `AuthCert`.

`ListOfHashes` contains the base64 encoding of the list of hashes, the data format is seen in listing 9. The `Id` value in `ListOfHashes` must be unique. The `ListOfHashes` can contain a maximum of 200 entries per request.

`HashSignature` contains the signature with the authentication certificate over the value of `ListOfHashes` before base64 encoding. `HashSignatureMechanism` specifies the algorithm used to sign the hash value.

For possible values for `HashSignatureMechanism` see chapter 4.1.

Response

```
HTTP/1.1 200 OK
Content-Length: 2510
Content-Type: application/json; charset=utf-8

{
  "ListOfSignature": [
    { "Id": 0, "Signature": "AUHlAMbw+TD3E0o...jrZPPSA==" },
    { "Id": 1, "Signature": "5lIHk...azo+1zg==" },
    ...
    { "Id": 99, "Signature": "VzvV/Hm...xzwp5ant+A==" }
  ]
}
```

Listing 10: batch sign hash value response

3.5 Test call

This call serves only as a connection test and always returns with HTTP state 200 - OK if the server is available.

Request

```
GET /SealQualified/v1/Test
Host: ...
```

Listing 11: test request

Response

```
HTTP/1.1 200 OK
```

Listing 12: test response

4 Values and constants

4.1 hash signature mechanism

For `HashSignatureMechanism` the following values values are supported:

- SHA256withRSA
- SHA384withRSA
- SHA512withRSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

4.2 ECDSA signature format

The signature created by the A-Trust server is an ECDSA signature in the following format:

$$\begin{aligned} signature &= Base64\_url(\ R + \ S) \\ \text{where } R &= \text{first coordinate of ECDSA point} \\ S &= \text{second coordinate of ECDSA point} \end{aligned}$$

5 Live and test system

5.1 Live-System

The live system for a.sign premium seal qualifiedis available via the following link:

URL: https://www.a-trust.at/SealQualified/v1

5.2 Test-System

For testing purpose the following parameters can be used:

URL: https://hs-abnahme.a-trust.at/SealQualified/v1

The required authentication certificate is available on github https://github.com/A-Trust/ASignSealQualified including free sample code and test clients.

6 Errorcodes

The following error messages are returned by the A-Trust service

HTTP statuscode	**description**
200 - OK	success
400 - bad request	error parsing the request
404 - not found	no seal for the given serial number or authentication certificate found
422 - unprocessable entity	given `AuthSerial` and `AuthCert` do not match
500 - internal server error	internal error

Table 2: errorcodes

References

[Cou14] Council of European Union: *Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC*, 2014. https://eur-lex.europa.eu/eli/reg/2014/910/oj.