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

a.sign RK HSM
Basic/Advanced/Premium

Developer Manual

(english translation)

Version: 3.1
Date: 1. März 2022

Contents

Date	Rev	Autor	Changes
01.02.2022	3.1	Patrick Hagelkruys	add chapter 2.6
01.03.2022	3.0	Patrick Hagelkruys	Changes to chapter 2.9
13.02.2020	2.2	Patrick Hagelkruys	Fixed error in Session Sign JWS example
18.09.2017	2.1	Patrick Hagelkruys	Explanation REST URL
12.06.2017	2.0	Patrick Hagelkruys	Rename a.sign RK HSM Basic/Advanced/Premium
09.06.2017	1.7	Patrick Hagelkruys	Rename a.sign RK MOBILE to a.sign RK ONLINE.
06.07.2016	1.6	Patrick Hagelkruys	Changes of url in samples. Removed the V1 version of the interface from the documentation. Extended descriptions for live system.
04.07.2016	1.5	Patrick Hagelkruys	live system data Rename `algo` to `alg` Link to Activation Help
12.05.2016	1.4	Patrick Hagelkruys	Rename product
28.04.2016	1.3	Patrick Hagelkruys	Fix error in documentation session login
24.02.2016	1.2	Patrick Hagelkruys	Fix error in chapter title
22.02.2016	1.1	Patrick Hagelkruys	Certificate serial number in hex format
26.01.2016	1.0	Patrick Hagelkruys	Interface V2 Command to change password More accounts for test cases
04.01.2016	0.9	Ramin Sabet Patrick Hagelkruys	Internal review
23.12.2015	0.8	Patrick Hagelkruys	Create a session for faster signature calls Plaintext signature JWS signature use cases
10.12.2015	0.7	Patrick Hagelkruys	Creating a customer account now also returns the certificate data Removed algorithm from signature and added to certificate information.
26.11.2015	0.6	Patrick Hagelkruys	englisch Version PHP samples

Table 1: document history

1. Overview

1.1. Summary

This document describes the interface for the a.sign RK HSM, in the basic, advanced and premium product versions. These three versions differ in the amount of included signatures and signing speed. More information is available via http://www.a-trust.at/registrierkasse.

The A-Trust HSM is a signature server for creating digital signatures according to the Austrian cash register security regulation [Bun15]

2. Communication Interface

2.1. Overview

A REST interface (HTTP POST and HTTP GET) is used as the communication protocol for a.sign RK HSM. In the URL, the username must be specified.

2.2. Signature commands with/without sessions

For the signature commands two variants are available.

In the simple approach without sessions, the user name and password is passed in the signature call. From this values a key must be generated which takes about half a second to complete.

Therefore, a second variant of the signature call was implemented to increase the speed of execution. In this variant at first a session login is necessary, in this call the key derivation takes place (about half a second). This session login provides two data fields (`sessionid` and `sessionkey`) which are needed for the subsequent signature commands.

2.3. Signature commands without session

2.3.1. Create signature

The request for creating a signature is a HTTP POST which contains a JSON object. The JSON objects consists of the password and the data to be signed (plain text).

For this call the transmitted data to_be_signed is first hashed and afterwards the ECDSA key is applied. According to [Jon15, chapter 3.1] the hash algorithm suitable for the key length is selected. At the moment an ECDSA P-256 key is used, therefore a SHA-256 hash function will be used, this results in the JWS ES256 algorithm.

The response consists of the signature data as reqired in [Jon15, chapter 3.4] in the following format.

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

Request

```
POST /asignrkonline/v2/{username}/Sign HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 152

{
"password":"123456789",
"to_be_signed":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 1: signature request

Response

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

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

Listing 2: signature response

2.3.2. Create signature, pass hash value

This call is similar to the signature creation from chapter 2.3.1, but instead of the plain text data a hash value is passed. Therefore the client program has to hash the data. Which hash algorithm must be used depends on the key length of the certificate and is defined in the table in [Jon15, Kapitel 3.1]. The key length has to be extracted from the certificate, see chapter 2.5.

Request

```
POST /asignrkonline/v2/{username}/Sign/Hash HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 152

{
"password":"123456789",
"hash":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 3: signature request (hash)

Response

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

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

Listing 4: signature response (hash)

2.3.3. Create signature, plaintext data

This command is similar to the signature creation in Chapter 2.3.1, but without the base64 encoding of the plaintext data.
For the plaintext vale an example is given in Chapter 6.3.

Anfrage

```
POST /asignrkonline/v2/{Benutzername}/Sign/Plain HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 152

{
"password":"123456789",
"to_be_signed":"c2...WNl=.A43c...Kj="
}
```

Listing 5: signature request (plaintext)

Antwort

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

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

Listing 6: signature response (plaintext)

2.3.4. Create signature, JWS

The signature call is a HTTP POST with the contents of a JSON with password and data to be signed.

This call creates the JWS header in accordance with [Bun15, Anlage Z 13], the data supplied will be formated as descripted in [Jon15] and the entire JWS structure with signature will we returned.

Anfrage

```
POST /asignrkonline/v2/{Benutzername}/Sign/JWS HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 247

{
"password":"123456789",
"jws_payload":"_R1-AT0_DEMO-C...oRo="
}
```

Listing 7: signature request (JWS)

Antwort

```
HTTP/1.1 200 OK
Content-Length: 189
Content-Type: application/json; charset=utf-8

{
"result":"ey...J9.X1I...z0=.an...Q==",
}
```

Listing 8: signature response (JWS)

2.4. Signature commands with session

2.4.1. Session login

The session login consists of a PUT request which contains a JSON structure with password. The response contains a session id and a sessionkey, these are required for the subsequent signature commands.

A session is valid for one hour. With each successful signature command the validity increases by one hour. At the latest at midnight a session is terminated.

Anfrage

```
PUT /asignrkonline/v2/Session/{Benutzername} HTTP/1.1
Content-Type: application/json
Host: ...
```

```
Content-Length: 30

{
"password":"123456789"
}
```

Listing 9: session login request

Antwort

```
HTTP/1.1 200 OK
Content-Length: 120
Content-Type: application/json; charset=utf-8

{
"sessionid":"ervhiklakgcmifzgeuwwfuttsalffovl",
"sessionkey":"ak3k39oVApkGfZ92FhcbFmL38zK6EMunu4ooqh3foGY="
}
```

Listing 10: session login response

2.4.2. Session logout

This command is used to prematurely terminate a session.

Anfrage

```
DELETE /asignrkonline/v2/Session/{sessionid} HTTP/1.1
Host: ...
```

Listing 11: session logout request

Antwort

```
HTTP/1.1 200 OK
Content-Length: 16
Content-Type: application/json; charset=utf-8

{"result":true}
```

Listing 12: session logout response

2.4.3. Create signature (session)

This command corresponds to that in chapter 2.3.1, instead of user name and password and sessionid and sessionkey is passed.

Anfrage

```
POST /asignrkonline/v2/Session/{sessionid}/Sign HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 152

{
"sessionkey":"123456789",
"to_be_signed":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 13: signature request

Antwort

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

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

Listing 14: signature response

2.4.4. Create signature (session), pass hash value

This command corresponds to that in chapter 2.3.2, instead of user name and password and sessionid and sessionkey is passed.

Anfrage

```
POST /asignrkonline/v2/Session/{sessionid}/Sign/Hash HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 180

{
"sessionkey":"ak3k39oVApkGfZ92FhcbFmL38zK6EMunu4ooqh3foGY=",
"hash":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 15: signature request (Hash)

Antwort

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

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

Listing 16: signature response (Hash)

2.4.5. Create signature (session), plaintext

This command corresponds to that in chapter 2.3.3, instead of user name and password and sessionid and sessionkey is passed.

Anfrage

```
POST /asignrkonline/v2/Session/{sessionid}/Sign/Plain HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 152

{
"sessionkey":"123456789",
"to_be_signed":"c2...WNl=.A43c...Kj="
}
```

Listing 17: signaturs request (Plain)

Antwort

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

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

Listing 18: signature response (Plain)

2.4.6. Create signature (session), JWS

This command corresponds to that in chapter 2.3.4, instead of user name and password and sessionid and sessionkey is passed.

Anfrage

```
POST /asignrkonline/v2/Session/{sessionid}/Sign/JWS HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 247

{
"sessionkey":"123456789",
"jws_payload":"_R1-AT0_DEMO-C...oRo="
}
```

Listing 19: signature request (JWS)

Antwort

```
HTTP/1.1 200 OK
Content-Length: 189
Content-Type: application/json; charset=utf-8

{
"result":"ey...J9.X1I...z0=.an...Q==",
}
```

Listing 20: signature response (JWS)

2.5. Request certificate information

The call for reading the certificate information is a HTTP GET request.

Request

```
GET /asignrkonline/v2/{username}/Certificate HTTP/1.1
Host: ...
```

Listing 21: certificate information request

Response

```
HTTP/1.1 200 OK
Content-Length: 1540
Content-Type: application/json; charset=utf-8

{
"Signaturzertifikat":"MIIE...QA6o=",
"Zertifizierungsstellen":["MII...WSF"],
"Zertifikatsseriennummer":"963244432",
"ZertifikatsseriennummerHex":"3969F190",
"alg":"ES256"
}
```

Listing 22: certificate information response

The first two lines of the response are encoded in the required format for the „Beleg Gruppe“ (see RKSV – Anlage Detailspezifikation -Z6)

2.6. Request certificate information incl History

The call for reading the certificate information including historic certificates is a HTTP GET request.

Anfrage

```
GET /asignrkonline/v2/{Benutzername}/Certificates HTTP/1.1
Host: ...
```

Listing 23: certificate information incl. history request

Antwort

```
HTTP/1.1 200 OK
Content-Length: 3204
Content-Type: application/json; charset=utf-8

{
  "alg": "ES256",
  "Signaturzertifikate": [
    {
      "Zertifikatsseriennummer": "987234644",
      "ZertifikatsseriennummerHex": "3AD80154",
      "Signaturzertifikat": "MIIEv...GM=",
      "Zertifizierungsstellen": ["MIIE...SF"]
  },
  {
      "Zertifikatsseriennummer": "987234666",
      "ZertifikatsseriennummerHex": "3AD8016A",
      "Signaturzertifikat": "MIIEv...GM=",
      "Zertifizierungsstellen": ["MIIE...SF"]
  }
  ]
}
```

Listing 24: certificate information incl. history response

2.7. Request certification authority (ZDA) information

The call for reading the certificate authority information is a HTTP GET request.

Request

```
GET /asignrkonline/v2/{username}/ZDA HTTP/1.1
Host: ...
```

Listing 25: certificate authority information request

Response

```
HTTP/1.1 200 OK
Content-Length: 15
Content-Type: application/json; charset=utf-8

{
"zdaid":"AT1"
}
```

Listing 26: certificate authority information response

2.8. Password change

This command changes the password for the a.sign RK HSM basic/advanced/premium signing key.

Anfrage

```
POST /asignrkonline/v2/{Benutzername}/Password HTTP/1.1
Content-Type: application/json
Hostr: ...
Content-Length: 53


{
  "currentpassword":"123456789",
  "newpassword":"987654321"
}
```

Listing 27: change password request

Antwort

```
HTTP/1.1 200 OK
Content-Length: 15
Content-Type: application/json; charset=utf-8

{
  "result":true
}
```

Listing 28: change password response

2.9. Partner commands

2.9.1. Creating an a.sign RK HSM basic/advanced/premium (customer account)

To create a customer account a HTTP POST request is used. The request consists of the partner password (partner_password), the product version and the data for the classification key (equivalent to „Ordnungsbegriff" in RKSV). The response contains the access data for the created a.sign RK HSM account.

The classification key (e.g. VAT ID) must be specified in two values. To differentiate the three possible categories of the classification key an integer is used. A second value contains the data itself.

- classification_key_type=0; VAT-ID (ger: UID-Nummer);
  e.g.: classification_key = ATU12345678
- classification_key_type=1; Global Location Number (GLN);
  z.B.: e.g.: classification_key = 5012345000008
- classification_key_type=2;Tax authority number and tax number; (ger: Finanzamt- und Steuernummer)
  z.B.: classification_key=12345/1234

There are three different version of the a.sign RK HSM

- product_version=1; a.sign RK HSM basic
- product_version=2; a.sign RK HSM advanced
- product_version=3; a.sign RK HSM premium

Additionally an e-mail address is required. This is used for communication with the customer when, for example, the certificate expires. Either the customer or partner e-mail address can be entered here. It is recommended to use a group mailbox instead of personal e-mail addresses.

Request

```
POST /asignrkonline/v2/{partner_username}/Account HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 261

{
"partner_password": "partnerpwd",
"classification_key_type": 0,
"classification_key": "ATU00000000",
"email": "test@test.com",
"product_version":1
}
```

Listing 29: create customer account request

Response

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

{
 "username":"u123456789",
 "password":"123456789",
 "Signaturzertifikat":"MIIE...QA6o=",
 "Zertifizierungsstellen":["MII...WSF"],
 "Zertifikatsseriennummer":"963244432",
 "ZertifikatsseriennummerHex":"3969F190",
 "alg":"ES256"
}
```

Listing 30: create customer account response

2.9.2. List customer accounts

List all customer accounts for a partner account.

Anfrage

```
POST /asignrkonline/v2/{partner_benutzername}/Account/List HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 43

{
  "partner_password": "partnerpwd"
}
```

Listing 31: list customer accounts request

Antwort

```
HTTP/1.1 200 OK
Content-Length: 744
Content-Type: application/json; charset=utf-8

{
   "accounts":[
      {
         "username":"u123456789",
         "dienst_ablaufdatum":"2025-01-20␣20:16:43Z",
         "zertifikat_ablaufdatum":"2025-01-20␣20:16:43Z",
         "Zertifikatsseriennummer":"1896317797",
         "ZertifikatsseriennummerHex":"71078365",
         "classification_value":"Steuernummer:␣99123/4568",
         "product_version":0
      },
      {
         "username":"u987654321",
         "dienst_ablaufdatum":"2025-01-20␣15:25:59Z",
         "zertifikat_ablaufdatum":"2025-01-20␣15:25:59Z",
         "Zertifikatsseriennummer":"1864761062",
         "ZertifikatsseriennummerHex":"6F25FEE6",
         "classification_value":"UID:␣ATU00000000",
         "product_version":0
      }
   ]
}
```

Listing 32: list customer acocunts response

See Appendis B for an overview of possible values for `product_version`.

2.9.3. Query available credits

Query available credis per product version.

Anfrage

```
POST /asignrkonline/v2/{partner_benutzername}/Credits HTTP/1.1
Content-Type: application/json
Host: ...
Content-Length: 43

{
  "partner_password": "partnerpwd"
}
```

Listing 33: Query available credits request

Antwort

```
HTTP/1.1 200 OK
Content-Length: 744
Content-Type: application/json; charset=utf-8


{
   "credits":[
      {
         "product_version":0,
         "credits":2
      },
      {
         "product_version":1,
         "credits":6
      },
      {
         "product_version":2,
         "credits":0
      },
      {
         "product_version":3,
         "credits":0
      }
   ]
}
```

Listing 34: Query available credits response

See Appendis B for an overview of possible values for `product_version`.

2.9.4. Add „Sorglos Paket“ to account

Extension of the validity of the signature service by another 5 years.

This command can only be made if there is a remaining term of 1 month or at the latest 1 year after the certificate or signature service has expired.

This call does not make any changes to the certificate, the certificate validity and certificate serial number remain the same. You can continue to use the signature service without having to make any adjustments to the cash register. The extension of the signature service entitles you to issue a new certificate, this is described in chapter 2.9.5.

The `username` parameter specifies the customer account for which the signature service should be extended.

The `product_version` parameter specifies the credit to be used to extend the validity of the signature service. (see chapter B) A credit from the same or more expensive product version can be used. If a more expensive credit is used, the customer account will be switched to the more expensive product version.

Anfrage

```
POST /asignrkonline/v2/{partner_benutzername}/Service/Extend HTTP/1.1
 Content-Type: application/json
 Host: ...
 Content-Length: 94

 {
   "partner_password": "partnerpwd",
   "username": "u123456789",
   "product_version": 0
 }
```

Listing 35: add „Sorglos Paket“ request

Antwort

```
HTTP/1.1 200 OK
```

Listing 36: add „Sorglos Paket“ response

A description of the „Sorglos Paket“ can be found under the following link (german).

www.a-trust.at/downloads/Downloads/Unterrichtung/User%20Guide%20Online-Sorglos-Paket.pdf

2.9.5. Issue new certificate for account with „Sorglos Paket“

If a „Sorglos Paket“ has been add to a customer account (see chapter ??), the associated certificate can be reissued. This changes the certificate and the certificate serial number. The new certificate values must be stored in the cash register, and the certificate serial number in the signing input data has to be changed. The new certificate needs to be added to the cash register in Finanzonline, also changes to the DEP may be required.

Anfrage

```
POST /asignrkonline/v2/{partner_benutzername}/Certificate/New HTTP/1.1
 Content-Type: application/json
 Host: ...
 Content-Length: 62

 {
   "partner_password": "partnerpwd",
   "username": "u123456789"
 }
```

Listing 37: issue new certificate request

Antwort

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

{
 "Signaturzertifikat":"MIIE...QA6o=",
 "Zertifizierungsstellen":["MII...WSF"],
 "Zertifikatsseriennummer":"963244432",
 "ZertifikatsseriennummerHex":"3969F190",
 "alg":"ES256"
 "zdaid":"AT1"
}
```

Listing 38: isseu new certificate response

3. Live-System

The live system for the a.sign RK HSM basic/advanced/premium is available at the following link:

URL: https://rksv.a-trust.at/asignrkonline/v2/

An activation support document for issuing an a.sign RK HSM via webshop ist available at the following link.
http://www.a-trust.at/downloads/registrierkasse/asignRKOnline_Aktivierungshilfe.pdf

4. Testsystem

For testing purpose the following parameters can be used:

URL: https://hs-abnahme.a-trust.at/asignrkonline/v2/

username: u123456789

password: 123456789

partner username: partner4711

partner password: partnerpwd

4.1. More test cases

4.1.1. a.sign RK HSM - password blocked

username: u039193334

password: 60z7dx

4.1.2. partner account without credits

partner username: partnerNoCredits

partner password: partnerpwd

5. Implementation of the request

5.1. Request with curl (Windows)

```
curl.exe -o outputSign.txt --cacert cas.pem -X POST -H "Content-Type:␣
    application/json" -d @requestSign.json https://.../v2/u123456789/Sign
```

Listing 39: command line call for signature

```
{
"password":"123456789",
"to_be_signed":"c2FtcGxlIHRleHQgZ...WNl"
}
```

Listing 40: requestSign.json

```
{
"signature":"GkXDFLK3C...feJhPwAA==",
}
```

Listing 41: outputSign.txt

```
curl.exe -o outputSign.txt --cacert cas.pem -X POST -H "Content-Type:␣
    application/json" -d @requestSign.json https://.../v2/u123456789/Sign
```

Listing 42: command line call for reading certificate information

```
{
"Signaturzertifikat":"MIIE...QA6o=",
"Zertifizierungsstellen":["MII...WSF"],
"Zertifikatsseriennummer":"963244432",
"alg":"ES256"
}
```

Listing 43: outputCert.txt

```
curl.exe -o outputZDA.txt --cacert cas.pem -X GET
    https://.../v2/u123456789/ZDA
```

Listing 44: command line call for reading certificate authority information

```
{
"zdaid":"AT1"
}
```

Listing 45: outputZDA.txt

5.2. Request with C-Sharp

```
string URL = "https://.../v2/u123456789/Sign";
string request = @"{
""password"":""123456789"",
""to_be_signed"":""c2Ftc...SBzZXJ2aWNl""
}";
byte[] data = System.Text.UTF8Encoding.UTF8.GetBytes(request);


HttpWebRequest webRequest = (HttpWebRequest)WebRequest.Create(URL);
webRequest.Method = "POST";
webRequest.ContentType = "application/json";

webRequest.ContentLength = data.Length;
webRequest.GetRequestStream().Write(data, 0, data.Length);
HttpWebResponse webResponse = (HttpWebResponse)webRequest.GetResponse();

StreamReader reader = new StreamReader(webResponse.GetResponseStream(),
    System.Text.UTF8Encoding.UTF8);
string ResponseText = reader.ReadToEnd();
```

Listing 46: C# example

5.3. Request with Java

```
String urlStr = "https://.../v2/u123456789/Sign";
String request = "{\"password\":\"123456789\",␣
    \"to_be_signed\":\"c2FtcGxl...XJ2aWNl\"}";

URL url = new URL(urlStr);
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("POST");
conn.setDoOutput(true);
conn.setDoInput(true);
conn.setUseCaches(false);
conn.setAllowUserInteraction(false);
conn.setRequestProperty("Content-Type", "application/json");

OutputStream out = conn.getOutputStream();
Writer writer = new OutputStreamWriter(out, "UTF-8");
writer.write(request);
writer.close();
out.close();

if (conn.getResponseCode() != 200) {
  throw new IOException(conn.getResponseMessage());
}
```

```java
BufferedReader rd = new BufferedReader(new
    InputStreamReader(conn.getInputStream()));
StringBuilder sb = new StringBuilder();
String line;
while ((line = rd.readLine()) != null) {
  sb.append(line);
}
rd.close();
conn.disconnect();
String responseStr = sb.toString();
```

Listing 47: Java example

5.4. Request with PHP

```php
<?php
$url = 'https://.../v2/u123456789/Sign';
$data = '{"password":"123456789","to_be_signed":"c2FtcGx...ZXJ2aWNl"}';

$options = array(
    'http' => array(
        'header'  => "Content-type:␣application/json\r\n",
        'method'  => 'POST',
        'content' => $data,
    ),
);

$context  = stream_context_create($options);
$result = file_get_contents($url, false, $context);

var_dump($result);
?>
```

Listing 48: PHP example

Based on the stackoverflow answer [Bha15]

6. Usage of the different signature commands

The following sections describes the client sequence for preparing the data so that it can be passed correctly to the a.sign RK HSM. The following examples are given in pseudo-code.

6.1. Signatur

```
// generate JWS header
JWS_Protected_Header = Base64url(UTF8('{"alg":"ES256"}'))

// generate JWS payload
JWS_Payload = '_R1-AT0_DEMO-CASH-BOX817_83468_2015-11-25T19:20:10_0,00_0,00_0,00_0,00_0
     ,00_sqv3XHcI8mU=_-3667961875706356849_d3YUbS4CoRo='
JWS_Payload = Base64url(UTF8(JWS_Payload))

// generate data to be signed
to_be_signed = JWS_Protected_Header + '.' + JWS_Payload
to_be_signed = Base64(ASCII(to_be_signed))

// a.sign RK HSM /Sign command
JWS_Signature = ATrustRegMobile.Sign('user','pwd',to_be_signed)

Ergebnis = JWS_Protected_Header + '.' + JWS_Payload + '.' + JWS_Signature
```

6.2. Signatur Hash

```
// generate JWS header
JWS_Protected_Header = Base64url(UTF8('{"alg":"ES256"}'))

// generate JWS payload
JWS_Payload = '_R1-AT0_DEMO-CASH-BOX817_83468_2015-11-25T19:20:10_0,00_0,00_0,00_0,00_0
     ,00_sqv3XHcI8mU=_-3667961875706356849_d3YUbS4CoRo='
JWS_Payload = Base64url(UTF8(JWS_Payload))

// generate hash
to_be_signed = JWS_Protected_Header + '.' + JWS_Payload
hash = SHA256(ASCII(to_be_signed))
hash = Base64(hash)

// a.sign RK HSM /Sign/Hash command
JWS_Signature = ATrustRegMobile.SignHash('user','pwd',hash)

Ergebnis = JWS_Protected_Header + '.' + JWS_Payload + '.' + JWS_Signature
```

6.3. Signatur Plain

```
// generate JWS header
JWS_Protected_Header = Base64url(UTF8('{"alg":"ES256"}'))

// generate JWS payload
JWS_Payload = '_R1-AT0_DEMO-CASH-BOX817_83468_2015-11-25T19:20:10_0,00_0,00_0,00_0,00_0
     ,00_sqv3XHcI8mU=_-3667961875706356849_d3YUbS4CoRo='
JWS_Payload = Base64url(UTF8(JWS_Payload))

// generate data to be signed
to_be_signed = JWS_Protected_Header + '.' + JWS_Payload

// a.sign RK HSM /Sign/Plain command
JWS_Signature = ATrustRegMobile.SignPlain('user','pwd',to_be_signed)

Ergebnis = JWS_Protected_Header + '.' + JWS_Payload + '.' + JWS_Signature
```

6.4. Signatur JWS

```
// generate JWS payload
JWS_Payload = '_R1-AT0_DEMO-CASH-BOX817_83468_2015-11-25T19:20:10_0,00_0,00_0,00_0,00_0
     ,00_sqv3XHcI8mU=_-3667961875706356849_d3YUbS4CoRo='


// a.sign RK HSM /Sign/JWS command
Ergebnis = ATrustRegMobile.SignJWS('user','pwd',JWS_Payload)

// split result to protected header, payload und signature
JWS_Protected_Header = Ergebnis.Split('.').Index(0)
JWS_Payload = Ergebnis.Split('.').Index(1)
JWS_Signature = Ergebnis.Split('.').Index(2)
```

6.5. Session Handling

```
// generate session
sessionid, sessionkey = ATrustRegMobile.Login('user','pwd')

// preapre signature data
to_be_signed=...
Ergebnis = ATrustRegMobile.Session_Sign(sessionid, sessionkey, to_be_signed)


// close session
ATrustRegMobile.Logout(sessionid)
```

A. Return codes of the interface

HTTP 200 Success

HTTP 400 (invalid bas64) Base64 of the input data could not be converted

HTTP 401 (invalid username or password) username/session or password wrong

HTTP 403 (username/session blocked) username/session blocked

HTTP 401 (username/session expired) username/session expired

HTTP 403 (username/session not supplied) username missing or too short

HTTP 403 (password/sessionkey not supplied) password/sessionkey missing or too short

HTTP 500 general error

HTTP 500 (error loading signing key) error loading signing key

B. Product version

- product_version=0; a.sign RK Online
- product_version=1; a.sign RK HSM Basic
- product_version=2; a.sign RK HSM Advanced
- product_version=3; a.sign RK HSM Premium

References

[Bha15] Bhatia, Dhruv: *CURL-less method with PHP5*, 2015. https://stackoverflow.com/questions/5647461/how-do-i-send-a-post-request-with-php#6609181, besucht: 2015-11-25.

[Bun15] Bundesministers für Finanzen: *Verordnung des Bundesministers für Finanzen über die technischen Einzelheiten für Sicherheitseinrichtungen in den Registrierkassen und andere, der Datensicherheit dienende Maßnahmen (Registrierkassensicherheitsverordnung, RKSV)*, 2015. https://www.bmf.gv.at/steuern/RKSV.pdf, besucht: 2015-11-16.

[Jon15] Jones, M.: *JSON Web Algorithms (JWA)*. RFC 7518, May 2015. https://tools.ietf.org/html/rfc7518, besucht: 2015-11-25.