Public document

BORICA
BANK SERVICES

# INTEGRATION GUIDE
for
The Relying parties with Cloud Qualified Electronic Signature service
provided by BORICA

User authentication by signing electronic documents

CONTENTS

ACRONYMS

CA	Certification Authority
CRL	Certificate Revocation List
CQES	Cloud Qualified Electronic Signature
DN	Distinguished Name
EGN	Uniform civil number assigned to each Bulgarian citizen
LNC	Uniform civil number assigned to a foreigner living in Bulgaria
eIDAS	electronic Identification, Authentication and trust Services (EU Regulation 910/2014)
EU	European Union
HSM	Hardware Security Module
OCSP	On-line Certificate Status Protocol
PIN	Personal Identification Number
PKI	Public Key Infrastructure
QC	Qualified Certificate

For additional information related to this document, please contact the Provider at:

41 “Tsar Boris III” Blvd.
1612 Sofia
BORICA AD
Tel.: 0700 199 10
E-mail: info@borica.bg
Official Web site: www.b-trust.bg

1 Basic Scenario 1 – User authentication with signature using client identifier (asynchronous)

1.1 Step 1: Send sign request

The Relying party's signing application (CAS) specifies the recipient (sends information how to identify the client(and client certificate) of the signing request through the HEADER parameter rpToClientAuthorization. The following options are available:

- personalId: customer's national personal identifier(bulgarian EGN or personal identifier of foreigner(LNC)).
- certId: customer certificate's identifier. This identifier can be found in B-Trust MOBILE application - the second part of the number next to the name of the customer in CQES menu. For example if the information on the screen is IVAN IVANOV(11111-22222) then certId is 22222. CAS should request this information from the customer with B-Trust MOBILE.
- profileId - customer profile's identifier concatenated with OTP password(Authorization code). This information can be found in B-Trust MOBILE application - from menu CQES - button CODE for the corresponding certificate.
- clientToken - customer's client token. In that case the customer is already registered in CAS enetering his profileId and OTP(Authorization code). This is done through /auth function (clientAuthUsingPOST) of this API which returns the client token as a result.

rpToClientAuthorization:

- personalId:put egn (Bulgarian national id)
- profileId:032-552574:523112
- clientToken:TPC7416DC60EEEC8252E2531413010AD170
- certId:1234

URL: https://cqes-rptest.b-trust.bg/signing-api/v2/sign
METHOD: POST
REQUEST HEADERS

KEY	VALUE	MANDATORY FIELD
Accept-language	bg || en	false
relyingPartyID	123456789	true
rpToClientAuthorization	personalId:egn	true
accept	application/json	true
Content-Type	application/json	true
KEY	VALUE	MANDATORY FIELD
Accept-language	bg || en	false

relyingPartyID	123456789	true
rpToClientAuthorization	personalId:egn	true

REQUEST BODY

```
{
   "contents": [
    {
                    "hashAlgorithm": "SHA256", //Signature digest algorithm
                    "signatureType": "SIGNATURE", //Signature type algorithm. SIGNATURE: Specifies
that the result will be digital signature(encrypted digest with the private key)
                    "confirmText": "Confirm system login", //This parameter is used in order to determine
the dialog box(and the text in it) to confirm signing in B-Trust MOBILE
                    "contentFormat": "DIGEST", //Type of the content(in the ‘data’ parameter) that will be
signed. DIGEST: The content(in the ‘data’ parameter) that is send to be signed is digest(BASE64
encoded) of the document.
                    "data": "U29tZSBkYXRhIGluIGJhc2U2NCBlbmNvZGVkIGZvcm1hdA==", The content
that will be signed. Tthis content should be BASE64 encoded
                    "padesVisualSignature": false, //This parameter is used in order to specify if the
signature in PDF signed file should be visualized in the signed file
                    "toBeArchived": false    //This parameter is used in order to specify that the signed
documents should be archived in QLTPS(Qualified Long Term Preservation Service) archive
    }
  ],
  "payer": "RELYING_PARTY", //Who will be charged in order to pay for the sign operation
(Client(CLIENT) or Relying party(RELYING_PARTY))
  "isLogin": true, //Flag that specifies if the request is for login in relying party system
  "relyingPartyCallbackId":"3fb1fbd9-7979-4a68-b57b" //unique ID of the request in relying party
system
}
```

RESPONSE

As a result of this function CAS receives callbackId with which to check the status of the requested login signature.

RESPONSE HEADERS

KEY	VALUE
Content-Type	application/json

RESPONSE BODY

Status 202

{

"data": {

"callbackId": "3fa137c6-b70f-4e63-bb37-e8d715644740",//Callback ID of the signature request. This ID is used to check the status of the signature request with /sign/{callbackId} function

"validity": "2021-09-13T18:54:32.173+00:00"//End date of the validity of the signing request. Till this date the relying party can check the status of the signing request with /sign/{callbackId} function.

},

"responseCode": "ACCEPTED", //Response code (status of the response)

"code": "ACCEPTED", //Response code (status of the response)

"message": "The request has been accepted." //Response message. The message can be localized with ‘Accept-language’ header

1.2 Step 2: CAS uses /sign/{callbackId} function (getSignedResult) or /sign/rpcallbackid/{rpCallbackId} function (getSignedResultByRpCallbackId) to check the status of the requested document (using “Polling” mechanism)

URL: https://cqes-rptest.b-trust.bg/signing-api/v2/sign/{callbackId} or https://cqes-rptest.b-trust.bg/signing-api/v2/sign/rpcallbackid/{rpCallbackId} (if the CAS has specified relyingPartyCallbackId in sendSignRequest operation)
METHOD: GET
REQUEST HEADERS

KEY	VALUE	MANDATORY FIELD
Accept-language	bg || en	false
relyingPartyID	123456789	true
accept	application/json	true

REQUEST PARAMETERS

PATH:

- callbackId // Callback ID(request id) - result of the synchronous operation /sign request (sendSignRequest); or
- rpCallbackId // Relying party callback ID (request id in relying party system)

RESPONSE HEADERS

KEY	VALUE

Content-Type	application/json

RESPONSE BODY

Status code 206 – login request is not signed

{

"data": {

"cert": null, //The X509 certificate(BASE64 encoded) that is used to sign the request. It is returned only when contentFormat parameter is SIGNATURE

"signatures": [//A list with signatures and and requests' statuses. The list is ordered in order of the requested documents

{

"status": "IN_PROGRESS", //Signature response status that will is returned as result

"signature": null, //Contains the signature or reference the signed document. If the content signature type is SIGNATURE then the digital signature is returned in this field. In all other content signature types an ID of the signed document is returned. This document can be downloaded with getSignedContent operation.

"signatureType": null //Signature type of the result

}

]

},

"responseCode": "IN_PROGRESS", //Response code (status of the response)

"code": "IN_PROGRESS", //Response code (status of the response)

"message": "Sign request is in progress." //Response message. The message can be localized with 'Accept-language' header

}

Status code 200 – document is signed

{

"data": {

"cert": null, //The X509 certificate(BASE64 encoded) that is used to sign the request. It is returned only when contentFormat parameter is SIGNATURE

"signatures": [//A list with signatures and and requests' statuses. The list is ordered in order of the requested documents

```
        {
          "status": "SIGNED", //Signature response status that will is returned as result
          "signature": "154832", //Contains the signature or reference the signed document. If the
content signature type is SIGNATURE then the digital signature is returned in this field. In all other
content signature types an ID of the signed document is returned. This document can be
downloaded with getSignedContent operation.
          "signatureType": "XADES_BASELINE_LTA_ENVELOPING" //Signature type of the result
        }
      ]
  },
  "responseCode": "COMPLETED", //Response code (status of the response)
  "code": "COMPLETED", //Response code (status of the response)
  "message": "Sign request is completed." //Response message. The message can be localized
with 'Accept-language' header
}
```

Status code 400

```
{
    "code": "BAD_REQUEST",
    "message": "The request could not be understood by the server due to malformed syntax (invalid
request parameters)."
}
```

Status code 401

```
{
    "code": "UNAUTHORIZED",
    "message": "The request is unauthorized."
}
```

Status code 404

```
{
    "code": "NOT_FOUND",
    "message": "The server has not found the signed content."
```

}

Status code 500

{

"code": "ERROR",

"message": "Internal server error. The server encountered an unexpected condition which prevented it from fulfilling the request."

}

1.3 Step 3: Download the signed content

After the status of the request becomes with status SIGNED the CAS uses /sign/content/{contentId} to download the signature. This contentId is received in field SIGNATURE after execute /sign/{callbackId} function. The signature can be downloaded in the next 7 days.

URL: https://cqes-rptest.b-trust.bg/signing-api/v2/sign/content/{contentId}
METHOD: GET
REQUEST HEADERS

KEY	VALUE	MANDATORY FIELD
Accept-language	bg || en	false
accept	application/octet-stream	true
relyingPartyID	123456789	true

REQUEST PARAMETERS
PATH:

- id // Id that is received in field SIGNATURE after execute /sign/{callbackId} operation

RESPONSE HEADERS

KEY	VALUE
Content-Type	application/xml or application/pdf or application/octet-stream

RESPONSE BODY

Status code 200

Content of the signed document

Status code 400, 401, 404, 500

2 Basic Scenario 2 - User authentication with signature Electronic Signature QR code (asynchronous)

2.1 Step 1: Send sign request

CAS sends signing request to Cloud QES customer using /signviaqr function(sendSignRequestViaQRUsingPOST). If parameters qrHeight and qrWidth are not specified a link will be received. Otherwise a QR code image(BASE64 encoded) with the specified dimensions will be returned.

URL: https://cqes-rptest.b-trust.bg/signing-api/v2/signviaqr
METHOD: POST
REQUEST HEADERS

KEY	VALUE	MANDATORY FIELD
Accept-language	bg || en	false
accept	application/json	true
Content-Type	application/json	true
relyingPartyID	123456789	true

REQUEST BODY

- with image size

```
{
   "qrHeight": 200, //Returned QR code height
   "qrWidth": 200, //Returned QR code width
   "request": { //Content signing request
      "content": { //Content which will be signed by the customer
         "confirmText": "Confirm system login via qr", //This parameter is used in order to determine
the dialog box(and the text in it) to confirm signing in B-Trust MOBILE
         "contentFormat": "DIGEST", //Type of the content(in the ‘data’ parameter) that will be signed
         "data": "U29tZSBkYXRhIGluIGJhc2U2NCBlbmNvZGVkIGZvcm1hdA==", // This content
should be BASE64 encoded
         "fileName": "login.xml", //Name of the document(file) that will be signed
         "hashAlgorithm": "SHA256", //Signature digest algorithm
         "padesVisualSignature": false, //This parameter is used in order to specify if the signature in
PDF signed file should be visualized in the signed file
         "signatureType": "SIGNATURE", //Signature type algorithm
```

```
            "toBeArchived": false //This parameter is used in order to specify that the signed documents
should be archived in QLTPS(Qualified Long Term Preservation Service) archive

        },

        "relyingPartyCallbackId": 12264723, //ID of the request in relying party system

        "callbackURL": "https://borica.bg/callbackURL", //URL address of WS where the relying party
will be notified when the request is signed

        "payer": "RELYING_PARTY", //Who will be charged in order to pay for the sign operation
(Client(CLIENT) or Relying party(RELYING_PARTY))

        "isLogin": true //Flag that specifies if the request is for login in relying party system

    }

}
```

- <u>without image size</u>

```
{

"request": { //Content signing request

        "content": { //Content which will be signed by the customer

            "confirmText": "Confirm system login via qr", //This parameter is used in order to determine
the dialog box(and the text in it) to confirm signing in B-Trust MOBILE

            "contentFormat": "DIGEST", //Type of the content(in the ‘data’ parameter) that will be signed

            "data": "U29tZSBkYXRhIGluIGJhc2U2NCBlbmNvZGVkIGZvcm1hdA==", // This content
should be BASE64 encoded

            "fileName": "login.xml", //Name of the document(file) that will be signed

            "hashAlgorithm": "SHA256", //Signature digest algorithm

            "padesVisualSignature": false, //This parameter is used in order to specify if the signature in
PDF signed file should be visualized in the signed file

            "signatureType": "SIGNATURE", //Signature type algorithm

            "toBeArchived": false //This parameter is used in order to specify that the signed documents
should be archived in QLTPS(Qualified Long Term Preservation Service) archive

        },

        "relyingPartyCallbackId": 12264723, //ID of the request in relying party system

        "callbackURL": "https://borica.bg/callbackURL", //URL address of WS where the relying party
will be notified when the request is signed
```

"payer": "RELYING_PARTY", //Who will be charged in order to pay for the sign operation (Client(CLIENT) or Relying party(RELYING_PARTY))

"isLogin": true //Flag that specifies if the request is for login in relying party system

}

}

RESPONSE HEADERS

KEY	VALUE
Content-Type	application/json

RESPONSE BODY

Status code 202 – with image size

{

"data": {

"callbackId": " f0ee6d39-21b7-4830-9940-baf7b0abe919", //Callback ID of the signature request. This ID is used to check the status of the signature request with /sign/{callbackId} function

"qrImage": "/9j/4AAQSkZJRgABAgAAAQABAAD/wCen2m18m3/AOfdpF+ …=", //QR image in BASE64 encoded format with requested dimensions

"qrPlain": "", //The link that is encoded in the QR

"validity": "2020-01-06T20:58:24.620+0000" //Request validity

},

"responseCode": "ACCEPTED", //Response code (status of the response)

"code": "ACCEPTED", //Response code (status of the response)

"message": "The request has been accepted."//Response message. The message can be localized with ‘Accept-language’ header

}

Status code 202 – without image size

{

"data": {

"callbackId": "e7be8819-93ad-4654-b2f4-2d077f2e5e6b", //Callback ID of the signature request. This ID is used to check the status of the signature request with /sign/{callbackId} function

```
            "qrImage": "", //QR image in BASE64 encoded format with requested dimensions


            "qrPlain":
"cqesd://534b700f1a4d187d3796392499fc242436558fbcc5e991ee16984477e831191b", //The link
that is encoded in the QR

            "validity": "2020-01-06T21:10:02.533+0000" //Request validity

      },

      "responseCode": "ACCEPTED",

    "code": "ACCEPTED",

    "message": "The request has been accepted."

}
```

Status code 400 – when relyingPartyCallbackId is already used

```
{

      "code": "BAD_REQUEST",

      "message": "The request could not be understood by the server due to malformed syntax (invalid
request parameters)."

}
```

Status code 401, 404, 500

2.2 Step 2: CAS uses /sign/{callbackId} function (getSignedResult) or /sign/rpcallbackid/{rpCallbackId} function (getSignedResultByRpCallbackId) to check the status of the requested document (using “Polling” mechanism)

2.3 Step 3: Download the signed content – same as in Basic Scenario 1