Q:marketing Aktiengesellschaft

Düsseldorfer Straße 193

45481 Mülheim an der Ruhr

Telefon (02 08) 30 15 0

Telefax (02 08) 30 15 555

Internet: www.Qmarketing.de

DESTO/DEBJE

Certificate signing request

1 Requirements

The user has registered at the portal.

2 API endpoint

The API for certificate signing requests will be available under the following URL path:

/api/certificate/request

The API will only be available via HTTPS and will only accept the HTTP method POST.

2.1 Data format

The portal client must provide the API with the following data:

- Type of portal client

  The portal server must know of which type a portal client is.

  Therefore, the portal client has to supply it's type during the initial certificate signing request.

  The following identifiers will be valid:

  - com.abb.ispf.client.welcome.app
  - com.abb.ispf.client.welcome.gateway
  - com.abb.ispf.client.knx-security-panel.app
  - com.abb.ispf.client.knx-security-panel.gateway
- Base64 encoded version of the certificate signing request in PEM format (the certificate key and certificate signing request have to be generated on the portal client).
- Friendly name

The portal client must embed the required data in the following format (see data/example/csr.json):

```
{
    "client-type": "com.abb.ispf.client.welcome.gateway",
    "client-csr": "BASE64-ENCODED STRING CONTAINING THE CERTIFICATE SIGNING
REQUEST IN PEM-FORMAT",
    "client-name": "Busch-Welcome IP-Gateway summer cottage"
}
```

Because JSON doesn't support multiline strings for values, the certificate signing request must be encoded with Base64 (http://en.wikipedia.org/wiki/Base64).

Given, we have the following certificate signing request:

```
-----BEGIN CERTIFICATE REQUEST-----
MIICijCCAXICAQAwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgTClNvbWUtU3RhdGUx
ITAfBgNVBAoTGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZDCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAL3yT6B/2ayd2jabbArG8gtmARIm5hWggiaNEaZi
43edaxQbid+Cmx0VJd0N4ny5u+DbPStzM9EvkOGAQ4lYF6BXvyrQt0iHg4V7HenQ
Iy6IJEpHUnZ74t/ccxKE9ctLXZ9wY+05TjEt+cy8zbS+BgSce63G++bNABXppctl
hhiZsuspVNs38qkg442rqZC37f0oFA8fXbZ9SNMYStikO5En8sLlcKQO5T7r/h7/
6aRldA/eMAJYi3GXf4PURnytAD29Sq8u3W8jM35WQRHEzKhi8lnXpQrbVqYYDDdp
LI8bgS8oj61czGcojJQyWDb+vqr4A3PGMCbTsCxhEnFCWk8CAwEAAaAAMA0GCSqG
SIb3DQEBBQUAA4IBAQAXTg6FGs/EWaIY8Sduo+Al4IPppaHWQvDKwYhATTiE9mFz
hMDh7k6DBKkx6pwff48DMnA/EFDGybaHtv9L5wM/baa6XDXbe/AobTSLeJTSSxxd
gmFH4OQga74pyK55l7zkuRCddgEqpU36ZYR5Ii//tXfDqw6zqjmUZ+c0zkztmLSH
ItI7EgYADRdjKb1Xq9JK7F5QI+xeVZ++1/c/fkOo+GAILB/N1a/nJgbkdaWcTljx
xncuDQ/VcSXYdYWoRQx95kPTtveT1dSFTZ5mHl7Q1o509GPZUcQBdBWTnIl9sSHl
bTLGOINN9s7s837rN0/p6zeQvTAPOcSMv0KUHn5R
-----END CERTIFICATE REQUEST-----
```

The corresponding Base64 encoded version will look like this (newline characters may NOT be included):

```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJQ2lqQ0NBWElDQVFBd1JURUxN
QWtHQTFVRUJoTUNRVlV4RXpBUkJnTlZCQWdUQ2xOdmJXVXRVM1JoZEdVeApJVEFmQmdOVkJBb1RH
RWx1ZEdWeWJtVjBJRmRwWkdkcGRITWdVSFI1SUV4MFpEQ0NBU0l3RFFZSktvWklodmNOCkFRRUJC
UUFEZ2dFUEFEQ0NBUW9DZ2dFQkFMM3lUNkIvMmF5ZDJqYWJiQXJHOGd0bUFSSW01aFdnZ2lhTkVh
WmkKNDNlZGF4UWJpZCtDbXgwVkpkME40bnk1dStEYlBTdHpNOUV2a09HQVE0bFlGNkJYdnlyUXQw
aUhnNFY3SGVuUQpJeTZJSkVwSFVuWjc0dC9jY3hLRTljdExYWjl3WSswNVRqRXQrY3k4emJTK0Jn
U2NlNjNHKytiTkFCWHBwY3RsCmhoaVpzdXNwVk5zMzhxa2c0NDJycVpDMzdmMG9GQThmWGJaOVNO
TVlTdGlrTzVFbjhzTGxjS1FPNVQ3ci9oNy8KNmFSbGRBL2VNQUpZaTNHWGY0UFVSbnl0QUQyOVNx
OHUzVzhqTTM1V1FSSEV6S2hpOGxuWHBRcmJWcVlZRERkcApMSThiZ1M4b2o2MWN6R2NvakpReVdE
Yit2cXI0QTNQR01DYlRzQ3hoRW5GQ1drOENBd0VBQWFBQU1BMEdDU3FHClNJYjNEUUVCQlFVQUE0
SUJBUUFYVGc2RkdzL0VXYUlZOFNkdW8rQWw0SVBwcGFIV1F2REt3WWhBVFRpRTltRnoKaE1EaDdr
NkRCS2t4NnB3ZmY0OERNbkEvRUZER3liYUh0djlMNXdNL2JhYTZYRFhiZS9Bb2JUU0xlSlRTU3h4
ZApnbUZINE9RZ2E3NHB5SzU1bDd6a3VSQ2RkZ0VxcFUzNlpZUjVJaS8vdFhmRHF3Nnpxam1VWitj
MHprenRtTFNICkl0STdFZ1lBRFJkaktiMVhxOUpLN0Y1UUkreGVWWisrMS9jL2ZrT28rR0FJTEIv
TjFhL25KZ2JrZGFXY1RsangKeG5jdURRL1ZjU1hZZFlXb1JReDk1a1BUdHZlVDFkU0ZUWjVtSGw3
UTFvNTA5R1BaVWNRQmRCV1RuSWw5c1NIbApiVExHT0lOTjlzN3M4MzdyTjAvcDZ6ZVF2VEFQT2NT
TXYwS1VIbjVSCi0tLS0tRU5EIENFUlRJRklDQVRFIFJFUVVFU1QtLS0tLQ==
```

2.2 Sending the request

The portal client must authenticate itself against the API via HTTP digest authentication by using the portal users credentials. If the authentication fails, the server will respond with the HTTP status code 401 "Unauthorized".

The portal client must make a HTTP POST request. If the portal client uses a different HTTP method, the portal server will respond with the HTTP status code 405 "Method not allowed".

The portal client must send the generated XML data as request body to the API. If the validation of the sent request data fails, the portal server will respond with the HTTP status code 400 "Bad Request".

2.2.1 Reasons for the returned HTTP status code 400 "Bad Request"

- Provided request body cannot be parsed (invalid JSON, ...)
- Provided client type is unknown
- Provided certificate signing request cannot be read (e.g. is not encoded with Base64, is not in PEM format)

2.2.2 The portal server processes the request in the following order

- Generating a unique serial number for the certificate to be signed
- Inserting a new portal client into the database
- Assigning the newly generated portal client to the authenticated user
- Signing the CSR with the ISPf Root CA

If all went well, the portal server returns the generated certificate back to the portal client along with the HTTP status code 201 "Created" and the Content-Type "application/x-x509-user-cert".

If any of these steps failed, the portal server responds with the HTTP status code 500. The portal client should advise the portal user to retry the procedure later.

2.2.3 Summary of possible HTTP status codes

- 201 - Everything went well; certificate is returned in response body with mimetype "application/x-x509-user-cert"
- 400 - Bad request; request body was malformed
- 401 - Unauthorized; digest authentication was not successful
- 405 - Wrong HTTP method was used; method 'POST' must be used
- 500 - Something along the process steps went wrong

2.2.4 Example using cURL

```
curl -k --digest --user portal_username:portal_password -X POST --data-binary @data/example/csr.json https://testing.ispf.datadevelopment.de/api/certificate/request
```

cURL reads the file @data/example/csr.json in binary mode and sends it as the request body to the API.

3 No websocket counterpart

For simplification purposes, the API for certificate signing requests will only be available via HTTPS.

3.1.1 A websocket counterpart will not be provided for the following reasons

- Every portal client that will use Websockets over HTTPS already has the ability to do normal HTTPS requests
- Digest authentication against the portal user database only has to be built once (otherwise we not only have to implement it in PHP but also in JavaScript)