iSHARE
certificates cheat sheet

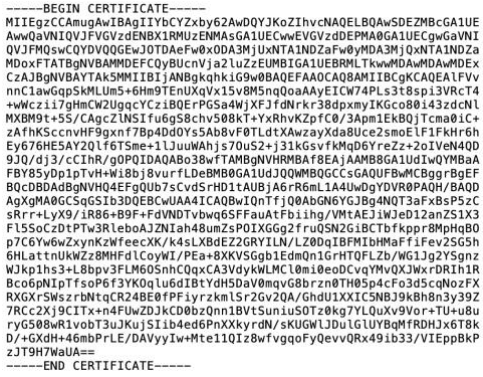
- **Certificates in general**
- **iSHARE specific certificate types**
- **OpenSSL conversion commands**

Certificates in general

A digital certificate, also known as a public key certificate, is used to cryptographically link ownership of a public key with the entity that owns it. Digital certificates are for sharing public keys to be used for encryption and authentication. Each public key has a corresponding private key, data encrypted using either one of the keys can only be decrypted with its counterpart. As the private key is kept secret, data that is decrypted using public key indicates that the source of the information can only be the holder of the private key. Digital certificates include the public key being certified, identifying information about the entity that owns the public key, metadata relating to the digital certificate and a digital signature of the public key created by the issuer of the certificate (1). Certificates come in various formats, of which the common ones are described below (2).

PEM Format

The PEM format is the most common format that Certificate Authorities issue certificates in. PEM certificates usually have extentions such as .pem, .crt, .cer, and .key. They are Base64 encoded ASCII files and contain "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----" statements. Server certificates, intermediate certificates, and private keys can all be put into the PEM format.
Apache and other similar servers use PEM format certificates. Several PEM certificates, and even the private key, can be included in one file, one below the other, but most platforms, such as Apache, expect the certificates and private key to be in separate files.



1 PEM format certificate

DER Format

The DER format is simply a binary form of a certificate instead of the ASCII PEM format. It sometimes has a file extension of **.der** but it often has a file extension of **.cer** so the only way to tell the difference between a DER .cer file and a PEM .cer file is to open it in a text editor and look for the BEGIN/END statements. All types of certificates and private keys can be encoded in DER format. DER is typically used with **Java platforms**.

PKCS#7/P7B Format

The PKCS#7 or P7B format is usually stored in Base64 ASCII format and has a file extention of **.p7b or .p7c**. P7B certificates contain "-----BEGIN PKCS7-----" and "-----END PKCS7-----" statements. A P7B file only contains certificates and chain certificates, not the private key. Several platforms support P7B files including **Microsoft Windows and Java Tomcat**.

2 Part of P7B format certificate

PKCS#12/PFX Format

The PKCS#12 or PFX format is a binary format for storing the server certificate, any intermediate certificates, and the private key in one encryptable file. PFX files usually have extensions such as **.pfx and .p12**. PFX files are typically used on **Windows machines** to import and export certificates and private keys. Please note that for Java the PKCS12 format is now recommended as industrial standard, removing the need for creating a Java key store (JKS).

(1) https://searchsecurity.techtarget.com/definition/digital-certificate
(2) https://www.sslshopper.com/ssl-converter.html

iSHARE specific certificate types

Within iSHARE, certificates and public keys are used for 3 main purposes. Identification & authentication of a legal entity, digital signatures to protect the integrity of digital claims and securing HTTP connections to servers. As specific internal usage and storage of certificates and key pairs is very much dependant on the on-premise situation, this section details only the standardised usage of certificates regarding iSHARE. This entails API HTTP requests, and certificate details found in these requests.

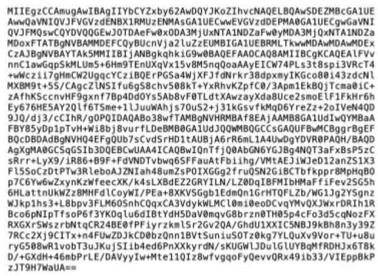
On certificates itself, the common X.509 format is used, where the intended key usage contains 'Digital Signature'. As for certificate authorities (CA), iSHARE trusts CAs that are certified to issue certificates within the eIDAS (3) framework. More specifically, only so-called 'qualified seals' within the scope of eIDAS. Within the Netherlands, during start-up phase 'PKI-Overheid Server certificates' will be allowed.

In general, the OAuth access token request is where certificates are used by a client as prove to their identity claim ("client_assertion"). The other main use is for signing server responses in the form of (signed) JSON Web Tokens. Please refer to https://jwt.io for everything related to JSON Web Tokens, and for libraries for various programming languages that provide functionality for creating, signing and validating JSON Web Tokens.

---

iSHARE JSON Web token 'x5c' header

When creating an iSHARE compliant JWT, the header field of this token contains an 'x5c' parameter. This field contains an array of strings. The value of the string is a PEM encoded certificate, without the BEGIN and END tags, as depicted on the right. The certificate used must match the certificate used by the client in the iSHARE admission process.



---

iSHARE Scheme Owner /testing/generate-jws

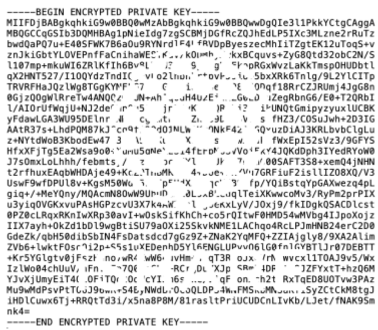
- certificateHash (request header)
  SAH256 hash (without ":") of certificate associated with private key
  Note: this API is only for testing and is not available for production, participants must be able to create their own JWS

7A6AC219F8F5D089E823D9B9263645DAC32B67846FFB5987C1590FE731E7CA87

---

iSHARE Scheme Owner /testing/generate-jws

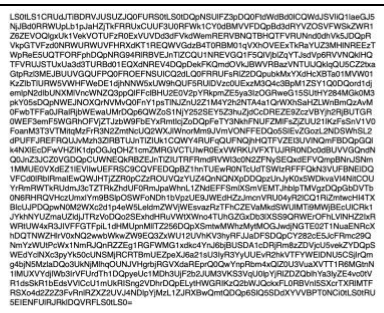
- privateKey (request body)
  raw 'text' of PEM formatted RSA private key
- Endpoint only accepts iSHARE Test certificates as input and will discard the private key after creating a signed JWT
  Note: this API is only for testing and is not available for production, participants must be able to create their own JWS



---

iSHARE Scheme Owner /certificate_validation

- certificate (request body)
  raw 'application/json', array with strings
- Each string is a PEM encoded certificate, which is then base64 UTF8 encoded
  Note: this API is for temporary use only, as it is recommended to do certificate validation yourself. The Scheme Owner is not liable for invalid responses from this endpoint



---

(3) https://ec.europa.eu/digital-single-market/en/trust-services-and-eid

OpenSSL conversion commands

Certificate formats can be converted into one another, enabling developers to acquire a suitable certificate format for their systems. This can be done via online tools or services, but also directly on your machine using OpenSSL. By using OpenSSL, you can keep your sensitive data on your machine, rather than sharing it online. After installing OpenSSL on your machine, you can use the following commands for certificate/key conversion.

Please refer to https://www.openssl.org to learn more about OpenSSL.

.p12 to .pem conversion

Convert .p12 (certificate+key) to .pem in new file

```
Openssl pkcs12 -in filename.p12 -out filename.pem
```

Convert .p12 (certificate+key) to .pem (certificate only) in new file

```
Openssl pkcs12 -in filename.p12 -out filename.pem -nokeys
```

Convert .p12 (certificate+key) to .pem (private key only) in new file

```
Openssl pkcs12 -in filename.p12 -out filename.pem -nocerts
```

.pem to .der conversion

Convert .pem (certificate) to .der (certificate) in new file

```
Openssl x509 -outform der -in filename.pem -out filename.der
```

Convert .der (certificate) to .pem (certificate) in new file

```
Openssl x509 -inform der -in filename.der -out filename.pem
```

.pem to .p7b conversion

Convert .pem (certificate) to .p7b certificate (requires CA certificate as well)

```
Openssl cr12pkcs7 -nocrl -certfile filename.pem -out filename.p7b -certfile
CAcertificatefilename.pem
```

Convert .p7b (certificate) to .pem (certificate)

```
Openssl pkcs7 -inform der -in filename.p7b -out filename.pem
```

.pem various actions

Inspect .pem (private key) cryptographic details, such as modulus and exponent, display in terminal

```
Openssl rsa -inform PEM -text -noout < filename.pem
```

Inspect .pem (certificate) SHA256 hash, display in terminal

```
Openssl x509 -in filename.pem -sha256 -noout -fingerprint
```