Exploit Certificates and Eliminate Tiresome Password Pains in z/OS and USS

Ulf Heinrich

SEGUS Inc

u.heinrich@segus.com

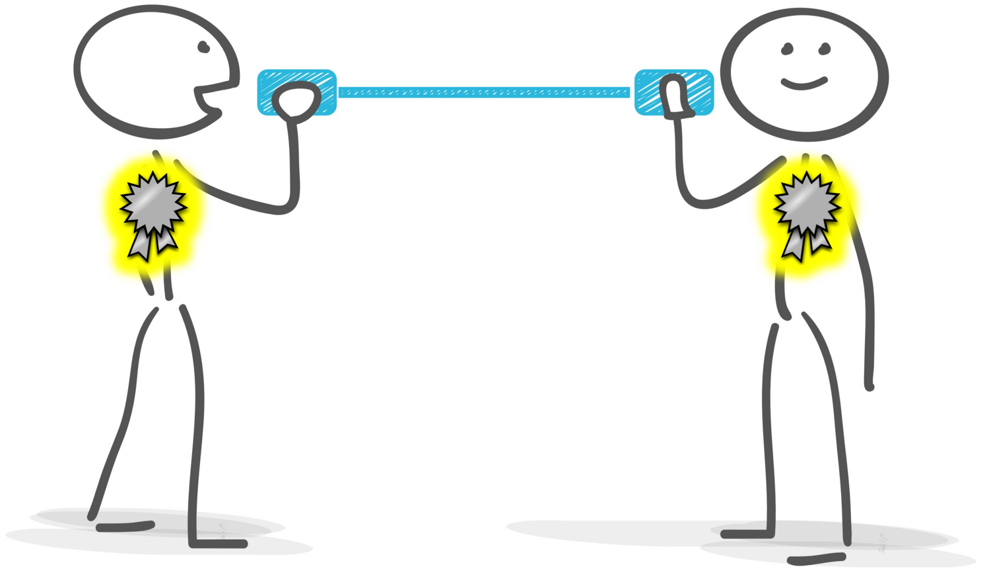
Agenda

- Digital Certificate Recap
- Certificate Lifecycle Management
- (Client) Certificate Authentication
    - Using Client Certificates
    - Using Distinguished Names
        - Issuer
        - Subject
        - Issuer + Subject
    - Real Examples from the ZOWE Ecosystem
        - as well as z/OSMF, UMS, SQLDI, Db2

Digital Certificate Recap

Secure (client – server) communication is based on X.509 certificates to:

1. Assure that a subject is really the one it claims to be.
2. Assure that the information exchanged isn‘t manipulated.
3. Assure that the communication is treated confidentially.



Digital Certificate Recap

Digital X.509 certificates are a common standard for decades and used in various areas:

websites

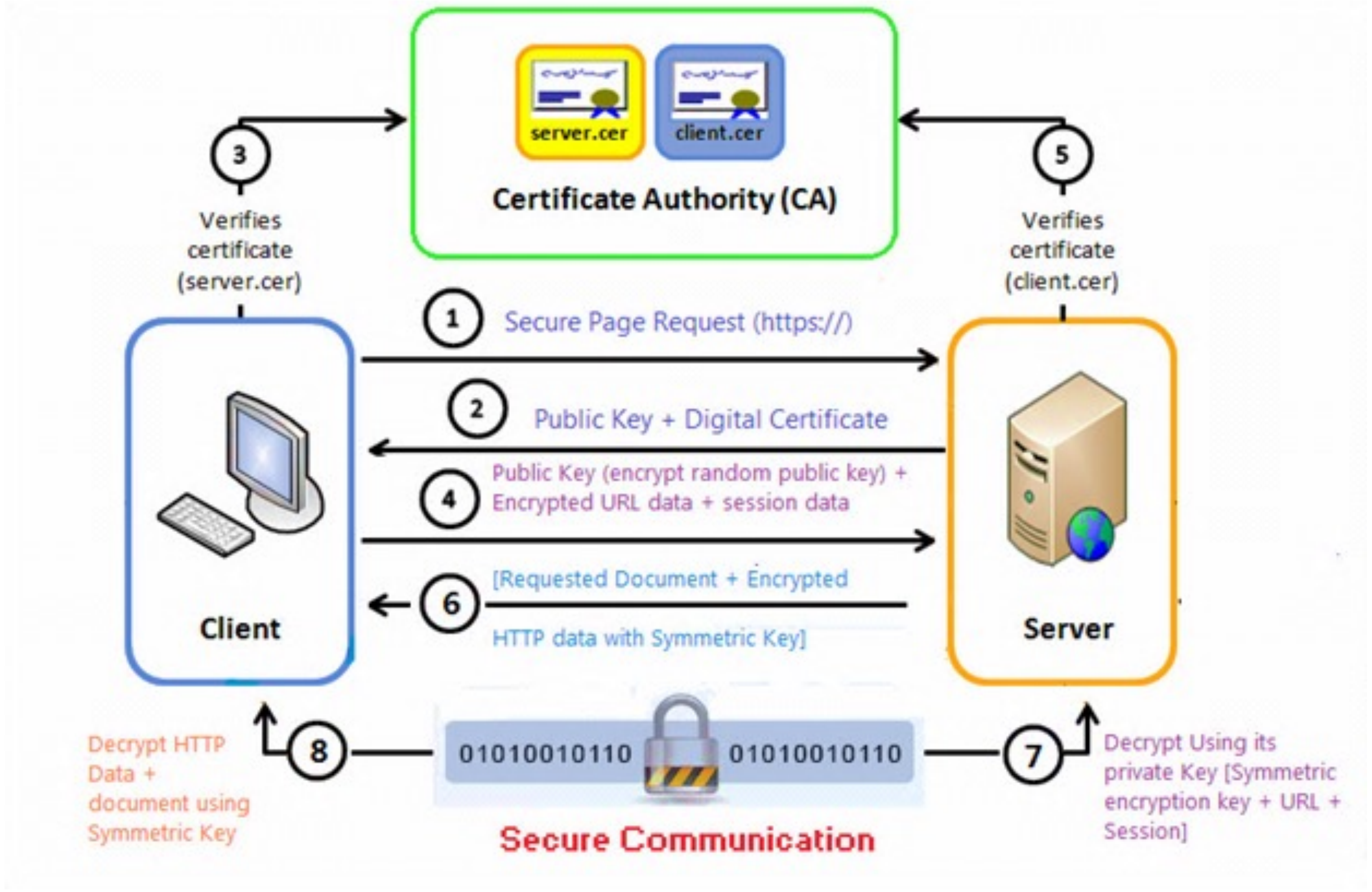
e-mails

software

documents

Digital Certificate Recap

TLS overview:


server.cer
client.cer
Certificate Authority (CA)
3
Verifies certificate (server.cer)
5
Verifies certificate (client.cer)
1
Secure Page Request (https://)
2
Public Key + Digital Certificate
4
Public Key (encrypt random public key) + Encrypted URL data + session data
6
[Requested Document + Encrypted HTTP data with Symmetric Key]
Client
Server
8
Decrypt HTTP Data + document using Symmetric Key
01010010110
01010010110
7
Decrypt Using its private Key [Symmetric encryption key + URL + Session]
Secure Communication

Digital Certificate Recap

- Certificates are stored either in
  - a KEYSTORE/TRUSTSTORE, or
  - RACF KEYRINGs

- Associated key pairs can be stored in
  - a data set
  - RACF
  - PKDS (ICSF PKA key data set)

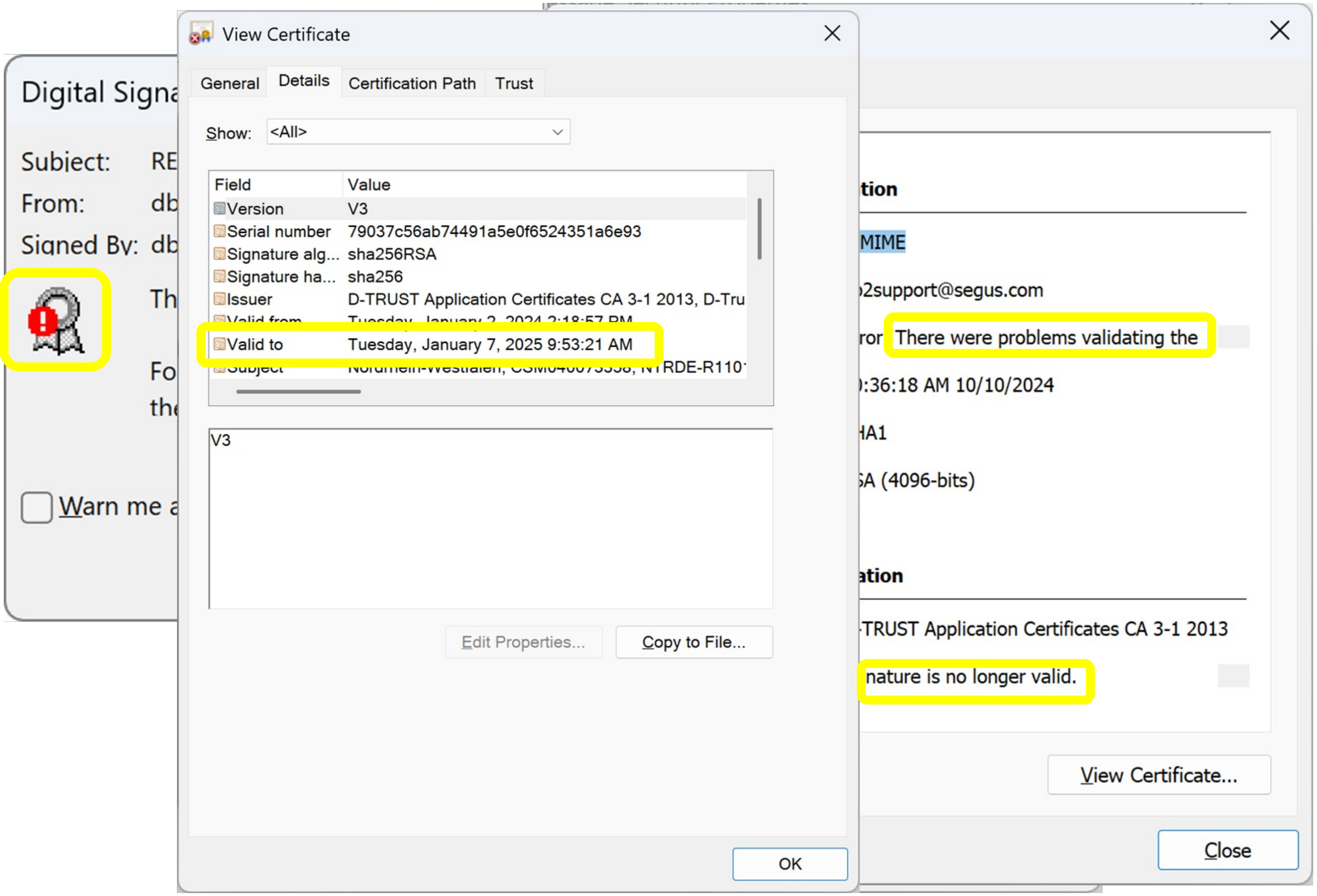
- Common tools are available to manage certificates
  - keytool
  - RACF/RACDCERT
    - PKDS option addresses the PKDS for key operations

Certificate Lifecycle Management

Like an identity card, certificates expire

- Certificate validity timeframe (NOTBEFORE – NOTAFTER) is shortened more and more
  - to reduce the risk of compromised certificates
  - ...and compensate unreliable revocation mechanisms
    - Online Certificate Status Protocol (OCSP)
    - Certificate Revocation List (CRL)
  - to force more frequent review/update of Subject Identity Information

Certificate Lifecycle Management



Digital Signa
Subject: RE
From: db
Signed By: db
Th
Fo
th
Warn me a

View Certificate
General | Details | Certification Path | Trust
Show: <All>

Field	Value
Version	V3
Serial number	79037c56ab74491a5e0f6524351a6e93
Signature alg...	sha256RSA
Signature ha...	sha256
Issuer	D-TRUST Application Certificates CA 3-1 2013, D-Tru
Valid from	Tuesday, January 2, 2024 2:18:57 PM
Valid to	Tuesday, January 7, 2025 9:53:21 AM
Subject	Nordrhein-Westfalen, CSM040073358, NTRDE-R110

V3

Edit Properties... Copy to File...
OK

tion
MIME
2support@segus.com
ror There were problems validating the
:36:18 AM 10/10/2024
HA1
A (4096-bits)
ation
TRUST Application Certificates CA 3-1 2013
nature is no longer valid.
View Certificate...
Close

Certificate Lifecycle Management

The maximum lifetime for a TLS certificate is continuously being reduced from 825 days to:

- 2020: max. 398 days
- March 15th, 2026: max. 200 days
- March 15th, 2027: max. 100 days
- March 15th, 2029: max: 47 days

However, a CA/Intermediate CA can still be up to 3650 days

→ Certificate lifecycle management is an important task and should be automated!

Certificate Lifecycle Management

- Certificates can be renewed, using the same SII and key pairs
- Certificates can be replaced, using updated SII and/or new key pairs
- A truststore/keyring can store many CAs and certificates as trusted entities, but a subject (e.g. server/service, or individual) can only use one, or another ID at a time!
  - → Make sure to change a certificate only after your clients trust it!
- Some servers/services require extra steps to “activate” a new certificate

Certificate Lifecycle Management

1. Generate the new CSR, certificate and/or key, or just renew an existing one:
   - RACDCERT GENREQ → new certificate request
   - RACDCERT REKEY → new private/public key pair
2. (Process CSR)
3. Renew/add the new certificate, or rollover to a new key pair
   - RACDCERT ADD → add the certificate for the USERID
   - RACDCERT REKEY and ROLLOVER → rekey a certificate
     - Consider RACDCERT ALTER to keep the original label
   - RACDCERT GENCERT → certificate renewal

Certificate Lifecycle Management

The easiest in-place renewal is a RACDCERT GENREQ that points to the current (expired) certificate

```
RACDCERT ID(<Certificate Owner>) GENREQ(LABEL('<Current
Certificate Label>’)) DSN('<OUTDSN>’)
```

→ Generates a new certificate request with exactly the same SII and key pair

```
RACDCERT ID(<Certificate Owner>) GENCERT('<INDSN>')
NOTAFTER(DATE(<New Expiration Date>)) SIGNWITH(CERTAUTH
LABEL('<Signing Certificate Authority>’))
```

→ Generates a new certificate, with a new expiration date, signed by the given CA

Certificate Lifecycle Management

- ZOWE requires the STC to be restarted to pick up the new certificate
  - That includes ZOWE apps, like Unified Management Server, Admin Foundation, …
- z/OSMF requires the STC to be restarted to pick up the new certificate
- SQLDI requires the STC to be restarted to pick up the new certificate
- Db2 requires DDF to be restarted, or a MODIFY REFRESH of the PAGENT
  - Data Sharing: DDF restart required
  - Non Data Sharing: PAGENT refresh required

Certificate Lifecycle Management

- How do connecting parties treat the updated/renewed certificate?
  - If it’s self signed?
  - If it’s CA signed?

  - If it trusts the subject’s certificate explicitly?
  - If it trusts the CA (issuer of the subject’s certificate)?

Certificate Lifecycle Management

- Certificate lifecycle management recommendations:
  - Make sure to use an internal, or external CA signing your certificates and trust it instead of an individual certificate!
  - Make sure to trust changed certificates before changing the server/service
  - Keep an old certificate/key in case you have encrypted content, like e-mails

Digital Certificate Recap

Secure client – server communication starts with a secure connection request, (e.g. https, ftps, ...) and often requires to specify a secure port:

https://s0w1.dus.seg.de:**10443**/zosmf

1. Connection request from a client to a server
2. Server replies with its UNIQUE certificate
3. Verification of the replying server and its trustworthiness by the client
4. Connection-dependent handshake of the encryption between client and server

Optionally: Certificate authentication of the client
Verification of the client by the server

5. Start encrypted communication

(Client) Certificate Authentication

An optional client certificate allows certificate-based client authentication, but where to get a client certificate from?

- → Generate them exactly like your ZOWE, z/OSMF, UMS, SQLDI, or Db2 server certificates (refer to last year's presentation for details and examples)
- → However, if you already have client certificates used to prove your identity (e.g. S/MIME, eID), you just need to make them known to your servers

(Client) Certificate Authentication

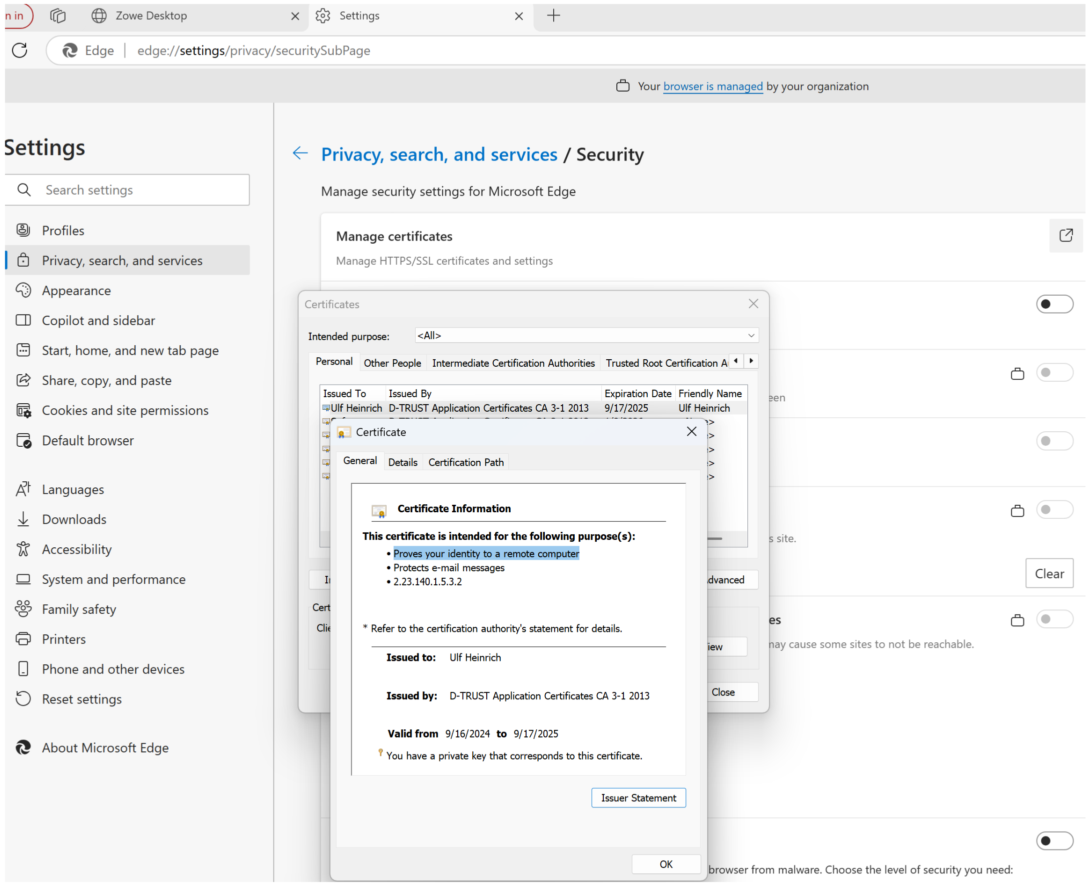
Once a certificate is generated/available, either

- associate it with a user ID, or
- refer to it, using Distinguished Names
  - Issuer
  - Subject
  - Issuer + Subject

→ Very flexible, especially from a lifecycle perspective

(Client) Certificate Authentication

- Modern applications are often accessed using a browser
  - A client certificate is stored in the client's private certificate store



Zowe Desktop | Settings

Edge | edge://settings/privacy/securitySubPage

Your browser is managed by your organization

Settings

Search settings

- Profiles
- Privacy, search, and services
- Appearance
- Copilot and sidebar
- Start, home, and new tab page
- Share, copy, and paste
- Cookies and site permissions
- Default browser
- Languages
- Downloads
- Accessibility
- System and performance
- Family safety
- Printers
- Phone and other devices
- Reset settings
- About Microsoft Edge

Privacy, search, and services / Security

Manage security settings for Microsoft Edge

Manage certificates

Manage HTTPS/SSL certificates and settings

Certificates

Intended purpose: <All>

Personal | Other People | Intermediate Certification Authorities | Trusted Root Certification A

Issued To	Issued By	Expiration Date	Friendly Name
Ulf Heinrich	D-TRUST Application Certificates CA 3-1 2013	9/17/2025	Ulf Heinrich

Certificate

General | Details | Certification Path

Certificate Information

This certificate is intended for the following purpose(s):

- Proves your identity to a remote computer
- Protects e-mail messages
- 2.23.140.1.5.3.2

* Refer to the certification authority's statement for details.

Issued to: Ulf Heinrich

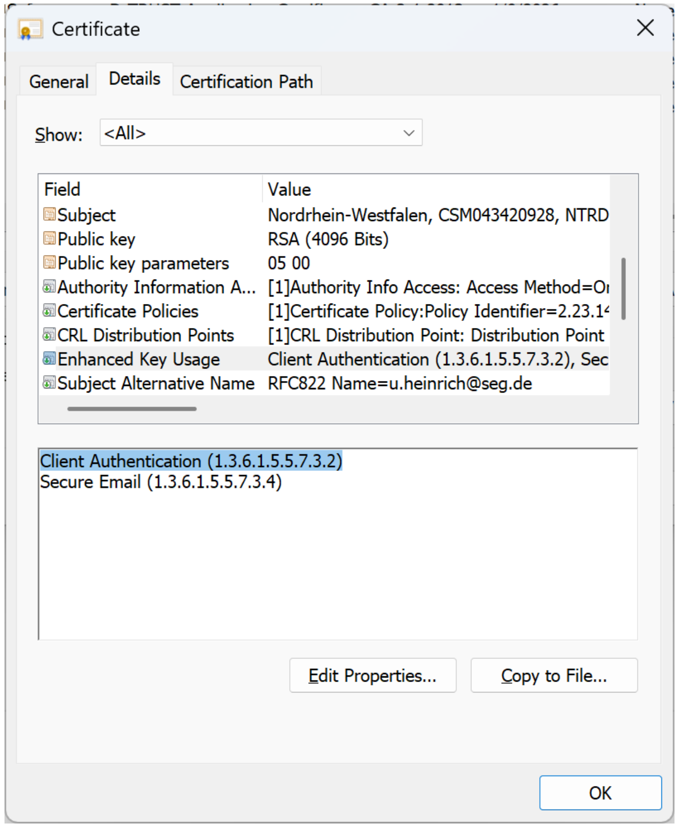
Issued by: D-TRUST Application Certificates CA 3-1 2013

Valid from 9/16/2024 **to** 9/17/2025

You have a private key that corresponds to this certificate.

(Client) Certificate Authentication

- Verify the certificate's object identifier (OID) for client authentication capabilities



Field	Value
Subject	Nordrhein-Westfalen, CSM043420928, NTRD
Public key	RSA (4096 Bits)
Public key parameters	05 00
Authority Information A...	[1]Authority Info Access: Access Method=Or
Certificate Policies	[1]Certificate Policy:Policy Identifier=2.23.14
CRL Distribution Points	[1]CRL Distribution Point: Distribution Point
Enhanced Key Usage	Client Authentication (1.3.6.1.5.5.7.3.2), Sec
Subject Alternative Name	RFC822 Name=u.heinrich@seg.de

```
Client Authentication (1.3.6.1.5.5.7.3.2)
Secure Email (1.3.6.1.5.5.7.3.4)
```

(Client) Certificate Authentication

- A client certificate becomes trusted if you trust the issuer (CA)
  - → Make sure the signing CA is added to your server's KEYRING/TRUSTSTORE
  - → Make sure that any intermediate CA is also added to your server's KEYRING/TRUSTSTORE
  - → Make sure the CAs are trusted
- Servers might treat client certificate authentication differently
  - Allow/deny connection
  - Map to a common USER ID (authorization)
  - Map to a specific USER ID (authorization)
  - Use the certificate as an additional level of authentication, instead of a USER ID/password replacement

Real Examples from the ZOWE Ecosystem

ZOWE supports X.509 client certificate authentication using either

- ZOWE API Mediation Layer (recommended default)
- ZOWE System Services (deprecated)

Real Examples from the ZOWE Ecosystem

- Enable X.509 client authentication within ZOWE's configuration YAML (default is disabled)

```
components:
gateway:
    enabled: true
    port: 7554
    debug: false
    apiml:
      security:
        auth:
          provider: zosmf
          zosmf:
            jwtAutoconfiguration: jwt
            serviceId: ibmzosmf
        authorization:
          endpoint:
            enabled: false
          provider: "native"
        x509:
          enabled: true
```

Real Examples from the ZOWE Ecosystem

- Choose between ZOWE's ML, or ZSS
  - ML:

```
components.gateway.apiml.security.useInternalMapper: true
```

  - ZSS:

```
components.gateway.apiml.security.zosmf.applid: IZUDFLT
```

- Check and map, or add your client certificate(s):

```
RACDCERT CHECKCERT('HEINRIC.CERT.PEM')

RACDCERT MAP ID(HEINRIC) -
SDNFILTER('CN=Ulf Heinrich.O=Software Engineering
GmbH.C=DE') -
WITHLABEL('CLT-CERT_HEINRIC')

RACDCERT ADD('HEINRIC.CERT.PEM') ID(HEINRIC) -
WITHLABEL('CLT-CERT_HEINRIC') TRUST

SETROPTS RACLIST(DIGTNMAP) REFRESH
```

Real Examples from the ZOWE Ecosystem

- Make sure you have an exact mapping of the subject‘s and/or issuer‘s DN

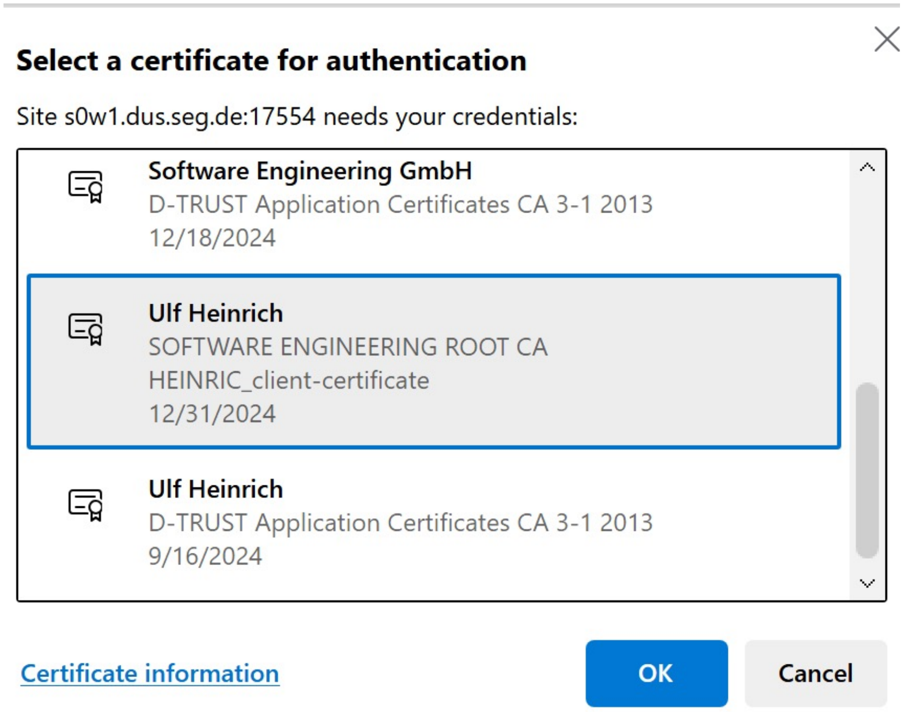
```
RACDCERT MAP ID(HEINRIC) -
SDNFILTER('CN=Ulf Heinrich.O=Software
Engineering GmbH.C=DE') -
WITHLABEL('CLT-CERT_HEINRIC')
```

- Consider using optional certificate models

```
RACDCERT ID(HEINRIC) MAP('HEINRIC.CERT.PEM')
WITHLABEL('CLT-CERT_HEINRIC') IDNFILTER('CN=')
TRUST
```

Real Examples from the ZOWE Ecosystem

- If a client has multiple certificates to choose from, you'll be prompted



Select a certificate for authentication

Site s0w1.dus.seg.de:17554 needs your credentials:

Software Engineering GmbH
D-TRUST Application Certificates CA 3-1 2013
12/18/2024

Ulf Heinrich
SOFTWARE ENGINEERING ROOT CA
HEINRIC_client-certificate
12/31/2024

Ulf Heinrich
D-TRUST Application Certificates CA 3-1 2013
9/16/2024

- BUT: always make sure that the issuer's (CA) certificate is trusted by ZOWE

Real Examples from the ZOWE Ecosystem

- Not only an interactive logon to the ZOWE Desktop allows client certificate authorization, but also services and apps:
  - e.g. CURL,:

```
curl -X POST \
--cert /path/to/mycert.pem \
--key /path/to/mykey.pem \
https://api-mediation-
layer:7554/gateway/api/v1/auth/login -v
```

  - or Java

    → refer to client-cert-auth-sample.jar sample of your ZOWE installation (/build/libs)

Real Examples from UMS and z/OSMF

- IBM Unified Management Server uses a DBA user ID and it can be authenticated by a client certificate
  - The KEYRING of the DBA ID can have the personal certificate used for client certificate authentication only
    - No ZOWE Server Certificate
    - No UMS Server Certificate
  - Multiple DBA user ID ←→ certificate associations possible
    - UMS default DBA: ZWESVUSR.KEYRINGA
    - Db2 specific UMS DBAs: ZWESVUSR.KEYRINGB

  → Set up via AT-TLS SAFCheck client authentication for Db2
- For z/OSMF, enable client certificate authentication

  → `SSL_CLIENT_AUTH=true`
- z/OSMF support both, client certificate authentication to
  - z/OSMFs REST services API
  - enable client certificate browser log in

Real Examples from SQLDI and Db2

- SQLDI (currently) doesn’t support client certificate authentication

- Db2 supports client certificate authentication as part of its AT-TLS setup of the PAGENT:
  - TTLSEnvironmentAction needs to be modified as follows:
    - `HandShakeRole Server`**`WithClientAuth`**
  - The configuration supports different levels of security:
    - Set TTLSEnvironmentAdvancedParms depending on your needs:
      - `ClientAuthType` **`Required`**

        Trusted issuer → add trusted CA to Db2 KEYRING
      - `ClientAuthType` **`SAFCheck`**

        Known subject → map certificate to RACF user
      - `ClientAuthType` **`SAFCheck + SERVAUTH`**

        Permitted user → define SERVAUTH class/profile