HCLSoftware

Leveraging Domino CertMgr

Engage Conference 2026

Daniel Nashed

Motivation for automated Certificate Management

by Nash!Com

> **TL;DR** — Certificate maximum lifetimes dropped to 200 days in March 2026 and will reach 47 days by 2029. At that frequency, manual renewal becomes operationally impossible. HCL Domino CertMgr automates issuance and renewal end-to-end. For everything outside Domino — NGINX, load balancers, and other services — Vault and srvguard close the distribution gap. Rotating the private key on every renewal cycle is the part most deployments have not solved yet.

The Clock Is Already Ticking

For years, TLS certificate management was something you dealt with once a year. Renew, deploy, forget about it for twelve months. That era is ending — not gradually, but on a fixed, published schedule that is already in motion.

In April 2025, the CA/Browser Forum passed **Ballot SC-081v3**, with every major browser (Apple, Google, Microsoft, Mozilla) voting yes and not a single no vote among certificate issuers. The ballot introduces a phased reduction in maximum TLS certificate validity:

Date	Maximum Validity
~~Until 2026-03-15~~	~~398 days~~
Now	**200 days**
2027-03-15	100 days
2029-03-15	**47 days**

Domino 12+ CertMgr Design Goals

- **Domino**
  - Domino CertMgr provides automation for Certificate Management since **Domino 12.0**
- **Design Goals**
  - **Simplify Domino certificate management**
  - **No external tools** like OpenSSL command line to create keys and convert certificates needed!
  - Replace difficult to handle *.kyr files with **standard *.pem format**
  - **Full Let's Encrypt® / ACME CA integration**
  - Simplified flows for external certificate authorities
  - **Domain wide secure and automated deployment for “TLS Credentials”**
  - Automated update of certificates including **automatic cache update** in internet server tasks
  - Support **ECDSA** in addition to **RSA**

Before Domino 12 *.kyr files, kyrtool & OpenSSL

- **Domino always used the *.kyr file format**
  - Old IBM format nobody else can read or write
  - The only tool available to read and write is “kyrtool”
    - Very flexible but command line driven -- **not always easy to handle**
    - Replaced old **certreq.nsf** database which wasn’t easy to use either

- **Creating keys and CSR required an external tool like “openssl” command line**
  - Very powerful, but also very cryptic tool with confusing command line for most admins
- ***.kyr files have a corresponding *.sth containing the encoded password**
  - Can be decoded with simple perl script
- **kyr file cache for internet processes always needed restart for any *.kyr change**

Automating Certificate Operations

- CertMgr has been designed using **open standards** and **automation** in mind

- CertMgr out of the box supports automated flows
  - But there is much more to discover

- The following slides explain CertMgr functionality to provide the base for Certificate operations
  - Explain standards, concepts, features
  - Explain the background for each functionality where it is relevant for automation
  - Point out in where automation can be used on top of the standard functionality

HCLSoftware

Certificates & Keys

Exportable private key:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIBBzBiBgkqhkiG9w0BBQ0wVTA0BgkqhkiG9w0BBQwwJwQQEMfk6vWgAuIF+NSj
uaWC9QICEAACASAwDAYIKoZIhvcNAgsFADAdBglghkgBZQMEASoEENsBxtoTWQEY
4nguFsOlbCcEgaBgGlbFuszmeRQW4loDxuWzMBMap+0U0Gc44KlStcMUfeuhnDax
LIlCtWMMIgvxIzD+0sSj74eG/CnNDoXlLOjKyR8TePzAymQ+jd98x70FBlUPZwNC
YtrL9+yth8j5EYMbCZz0E/Yux9BdLREKNA3idQZPuozqkK0Oq/lvZUrda5VanzYw
qLa7tUlBB5hbrMOUhU2JmxVsyiSlmI/6J/Ix
-----END ENCRYPTED PRIVATE KEY-----
```

Certificate chain:

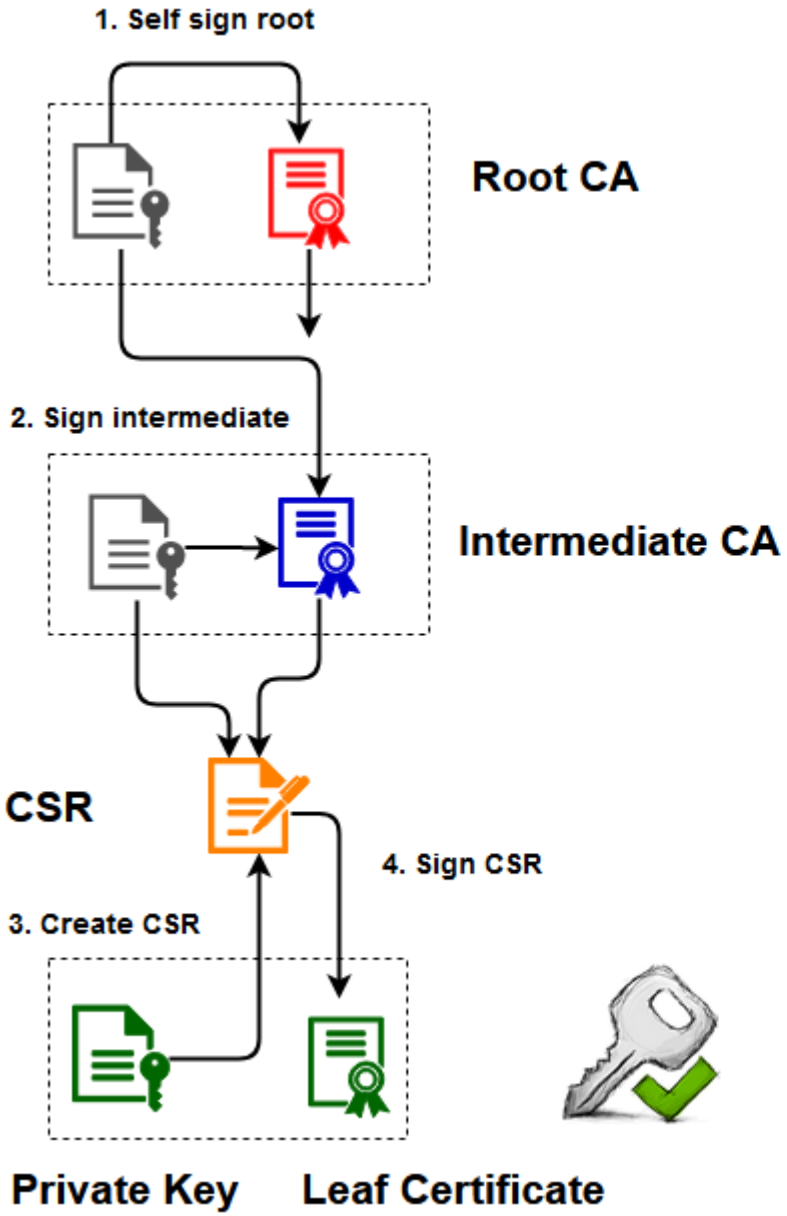
```
-----BEGIN CERTIFICATE-----
MIIBkTCCATegAwIBAgIQcr4pwloz4qjYKEu323bLqTAKBggqhkjOPQQDAjAfMRAw
DgYDVQQKDAdOYXNoQ29tMQswCQYDVQQDDAJDQTAeFw0yMTEyMDYxNjI2MTlaFw0y
MjEyMDgxNjI2MTlaMCsxEDAOBgNVBAoMB05hc2hDb20xFzAVBgNVBAMMDnd3dy5u
YXNoY29tLmRlMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEfOIGtztQHC1JB9oo
7fuB81lyuAFn0Grj6dYIXLHEQWIPP273E8XX+m8IWySpP5OlEt80hWw0eSGvmuRJ
HiEn56NJMEcwMAYDVR0RBCkwJ4IOd3d3Lm5hc2hjb20uZGWCFXd3dy54bi0tYmNo
ZXIta3ZhLmNvbTATBgNVHSUEDDAKBggrBgEFBQcDATAKBggqhkjOPQQDAgNIADBF
AiEAvgsuRQojEXd8V00eTIYHeoWINiVzvxl7vnLfQNPxY5cCIFDzE0n3xZE2+J6y
spEapub3+REXxBdLNU8sGA8ZziQ4
-----END CERTIFICATE-----
```

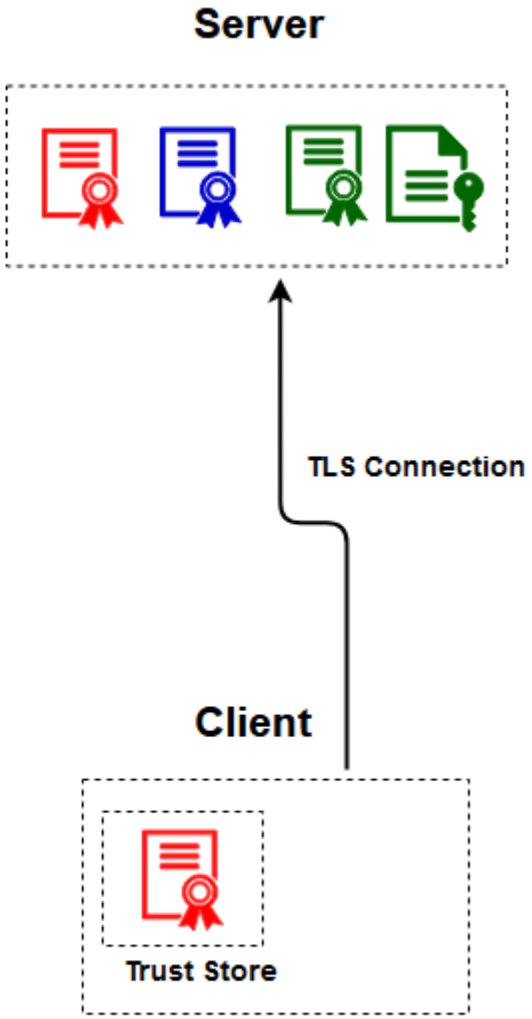
Private/Public keys

- **Private** keys are generated based on crypto graphic standards
- You can **derive** a public key from the **corresponding** private key
- The public key is shared with **anyone** who wants to encrypt for you
- The private key needs to be **always protected**!
- Private keys are also used to create signatures
  - For example in case of X.509 certificates private keys sign a CSR (certificate signing request)

- **Used for asymmetric (public/private key) encryption**
  - Most times for communication a symmetric session key is created for data encryption
  - Symmetric keys are encrypted with the asymmetric public key

CA and certificate chain





- CA is a self signed certificate installed as a trust into a **local trust store**

  CA & intermediate/sub CAs have a private key signed by next higher CA

  The last intermediate "**issuing CA**" signs CSR resulting in a **leaf certificate**
  - A CSR is signed by the private key with the information about what to certify (e.g. SAN, org)

  To verify the **certificate local root** from trust store + the full chain is used

PEM format

- Widely used de-facto standard to represent key and certificate information
  - https://en.wikipedia.org/wiki/Privacy-Enhanced_Mail
- **Text based format which can be merged and edited with any editor**
  - Base64 encoded form of the binary DER format + markers for begin and end of the data
- Supports unencrypted and encrypted keys
  - Encryption uses a “password/passphrase”
  - Data is still shown as text (Base64 encoded)
- Many applications use separate PEM files for key and certificate chain
  - Example: NGINX
- A big benefit:
  - Update certs without re-encrypting the key

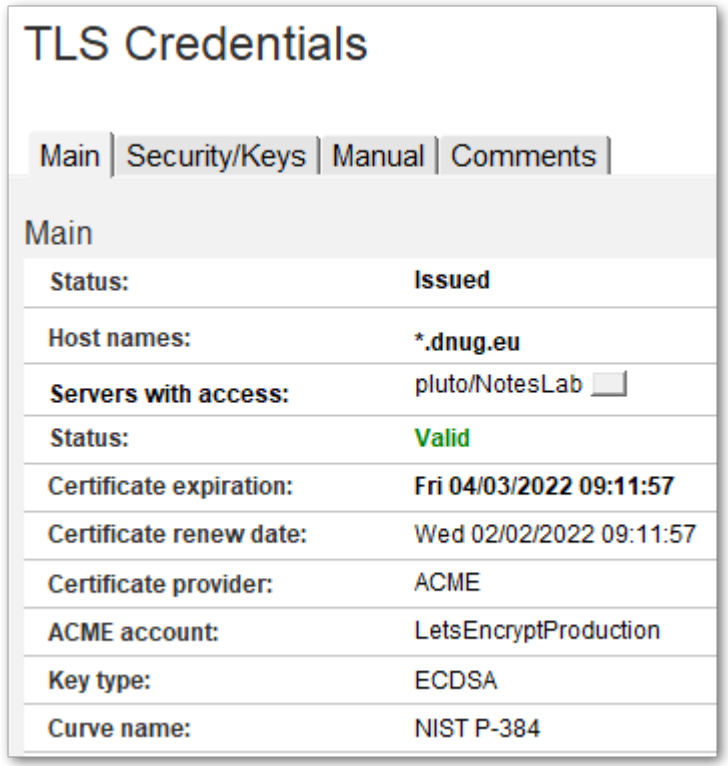
HCLSoftware

PKCS#12 (aka P12 or PFX)

- Widely used standard to store key and certificate in **one binary container**
  - https://en.wikipedia.org/wiki/PKCS_12
- **Binary format often encrypted with a password/passphrase**
  - Combining key and certificate chain in one encrypted file
- Often used by applications like HCL Sametime, HCL SafeLinx, Tomcat, etc
- PKCS#12 is also replacing the old legacy Java key store (JKS format)
- Examples: domino.p12, win.pfx
- Challenge
  - If password protected, you need to decrypt the p12 file before you can read anything
  - This makes certificate updates more challenging

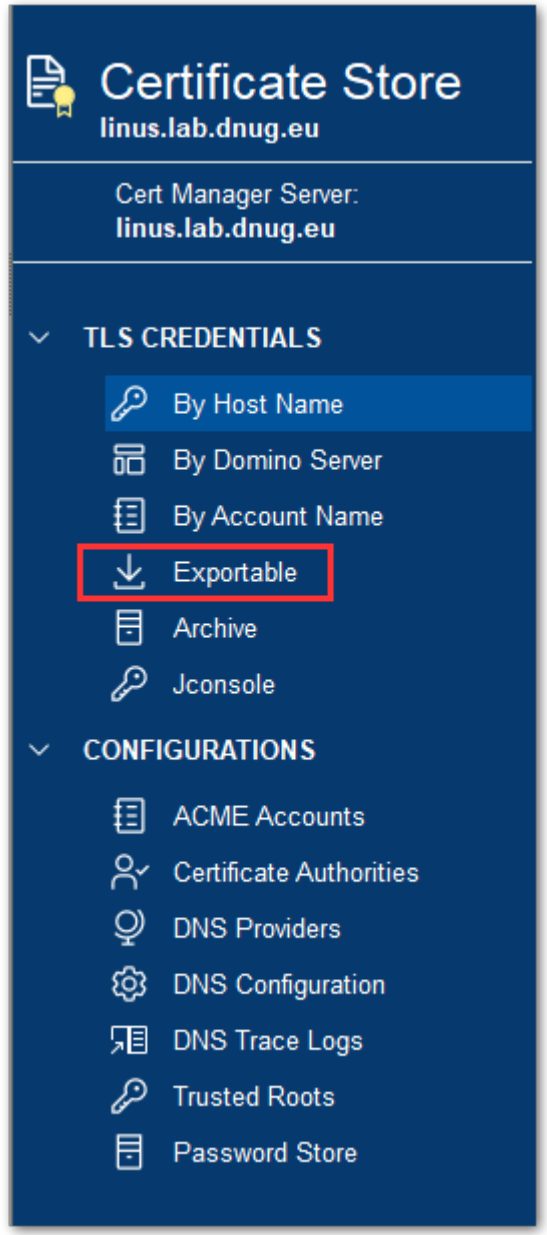
HCLSoftware

certstore.nsf



certstore.nsf

- **Domain wide database managed by CertMgr task**
- Secure, automated deployment for TLS Credentials and trusted roots
- **Private keys are encrypted with CertMgr server and the server specified in field “Servers with access:”**
  - Special designed Vault style encryption with new API
- Easy to use with modern interface
- CertMgr servertask is only supported on W64 and Linux64
  - AIX and OS400 can still leverage certstore.nsf and the new TLS Cache
    - Create replica manually



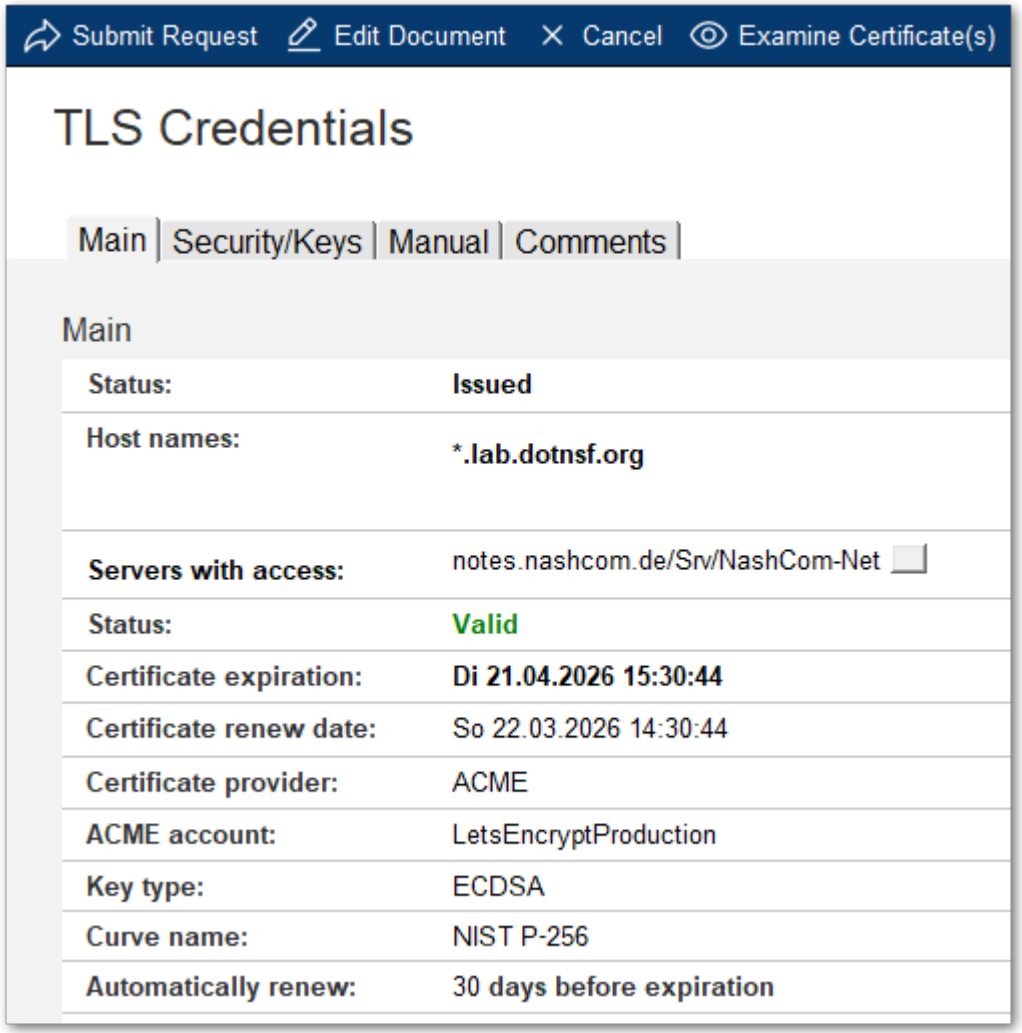
Create certstore.nsf on CertMgr Server

- **First server in domain starting the “certmgr” servertask is setup as the CertMgr Server**
  - Checks the **Domino directory profile on admin server** for an existing CertMgr server
  - If no server exists automatically creates the domain wide **certstore.nsf** database
  - Updates the directory profile on admin server to propagate the CertMgr server in the domain
  - **All operations are performed on CertMgr server**

- Starting the certmgr servertask on any additional server in the domain **creates a replica**
  - Each additional server acts like a “CertMgr client” and will just replicate the database every 2 minutes
  - Keeping the CertMgr servertask loaded is an optional convenience step
  - Any type of replication setup which ensures a short replication cycle can be used as well

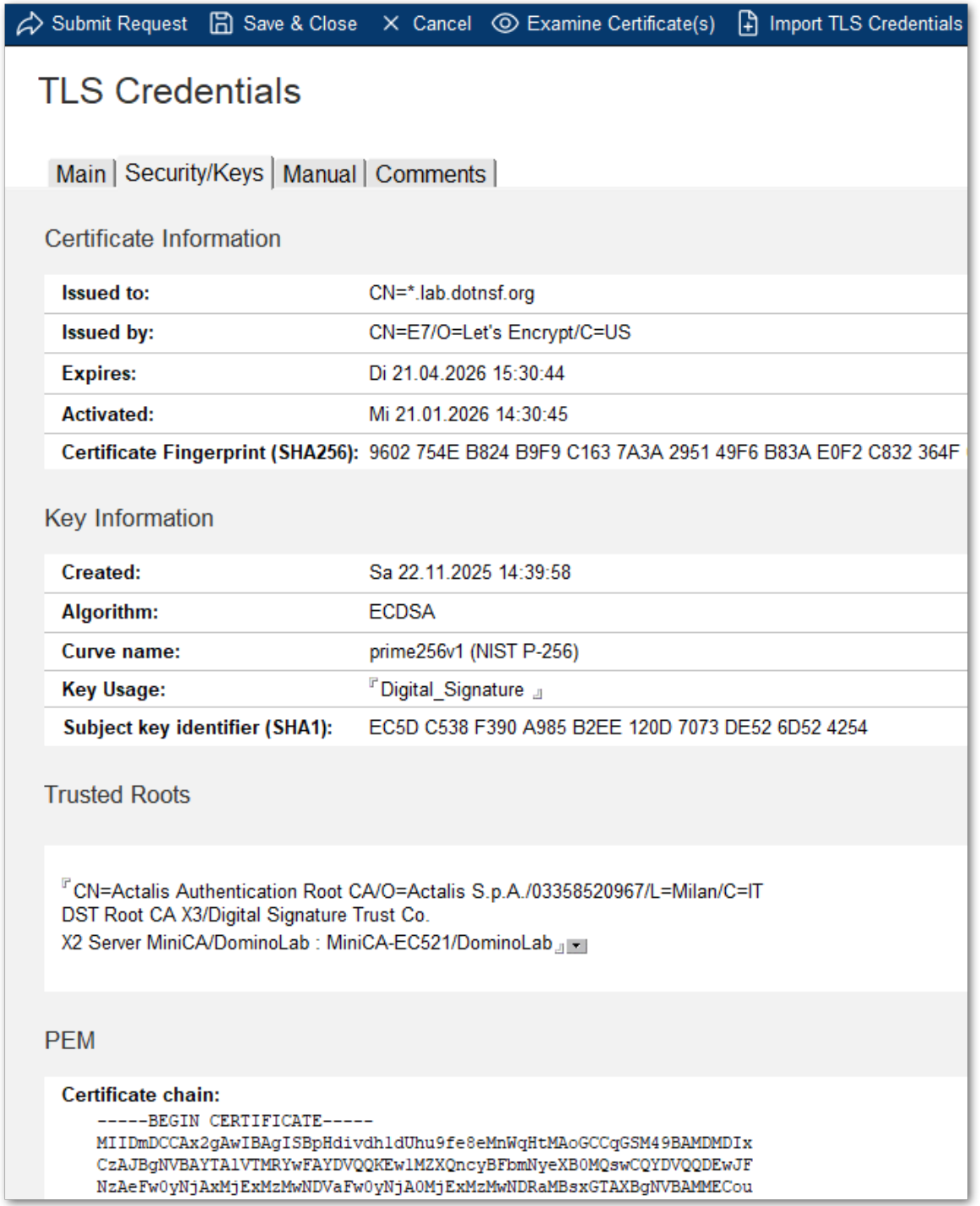
TLS Credentials

- **TLS Credentials** = private key + leaf certificate + chain (intermediates) + trusted root
- Replaces "***.kyr** files"
  - Stored in **PEM** format (text with base64 encoded data)
- Can be created via
  - Manual flows including import
  - ACME protocol (Let's Encrypt & others)
  - Micro CA
- Specify trusted roots used for client certificate verification
  - Used to be hidden in kyrfile.kyr and was difficult to manage



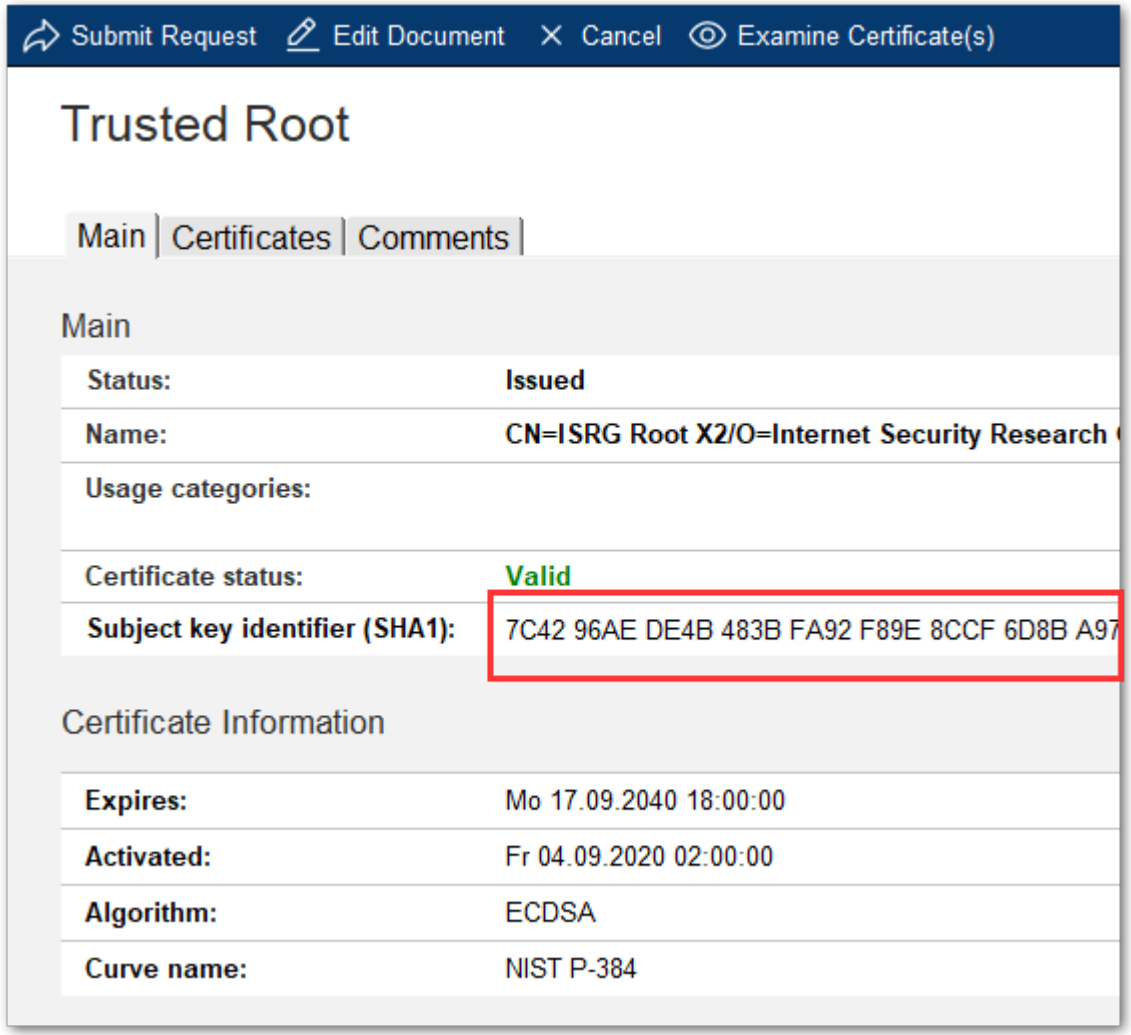
TLS Credentials – Details

- Details about certificates and key
- Issued to/by, expires, activated
- Key algorithm, bits/curve, usage
- Subject Key identifier used for lookups
- Trusted Roots trusted by this TLS Credentials
  - **Replaces imported roots in keyfile.kyr**
  - Much easier interface



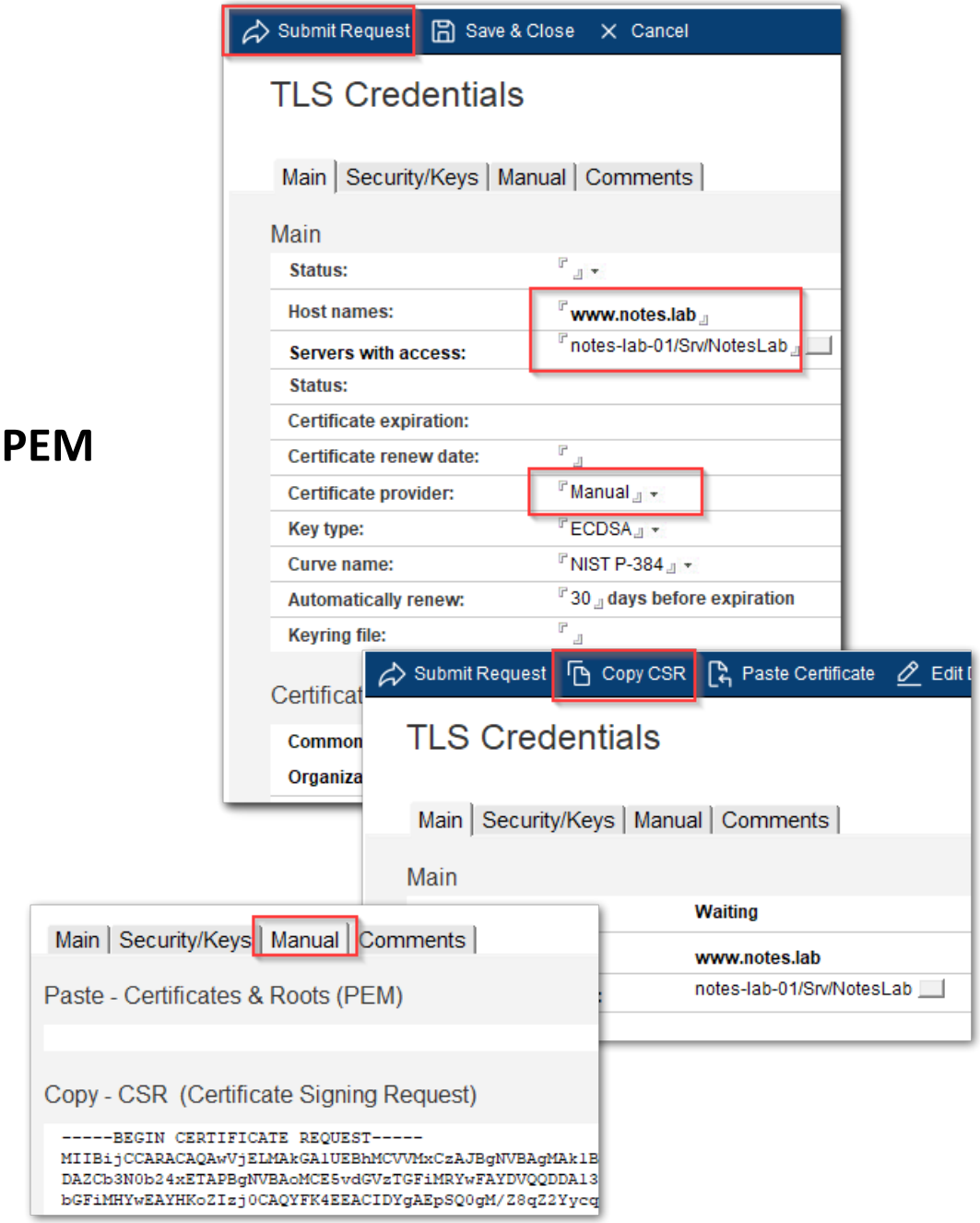
Trusted Roots

- Stored in trusted, secured certstore.nsf
  - Replicated domain wide
- The lookup identifier is “**Subject key identifier**”
- Auto complete certificate chains for all flows
  - Certificate chains are sorted & completed
  - Private Key → matching leaf certificate
  - intermediate certs in the right order
  - trusted root
- Tip: Intermediate certificates can be imported as “Trusted Root” to be used to complete chains
  - Just listed with warning, that they are no root



Manual Certificate Operations

- 1. CertMgr processes submitted requests and creates
  - Private key saved locally encrypted for **assigned servers**
  - **CSR** (**C**ertificate **S**igning **R**equest) signed by private key→ **PEM**
- 2. Admin sends **CSR** to **CA**
- 3. Admin imports certificate & chain (PEM) back
  - Paste full chain in any order and submits the form again
  - Duplicate certs are ignored
  - Missing intermediate certs and root are automatically added from “**Trusted Roots**” in **certstore.nsf**
- **Process can be automated using Lotus Script**
  - Certificates are stored in **PEM** = Text



HCLSoftware

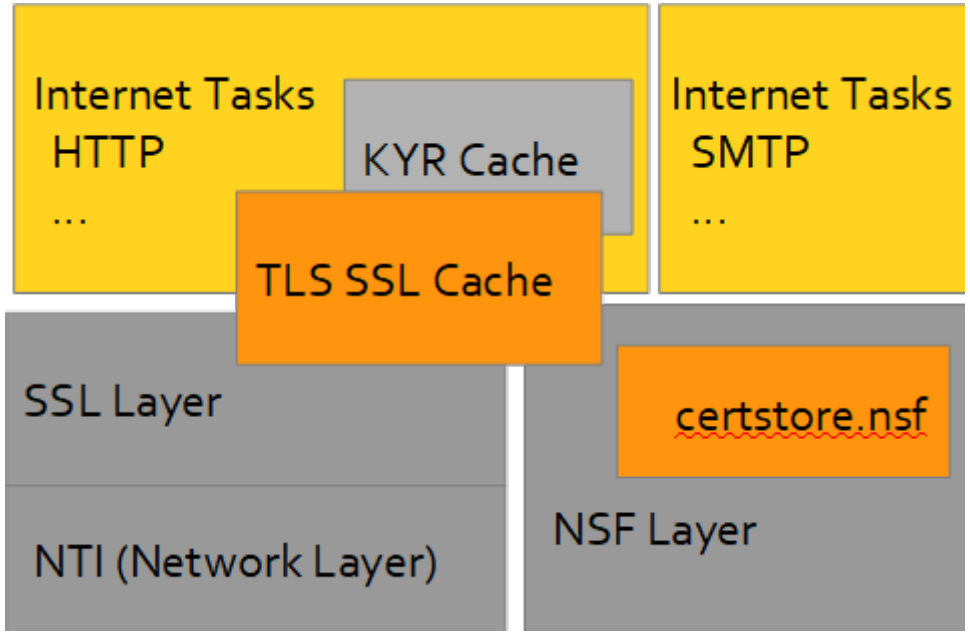
HCLSoftware

TLS Credential Cache

```
Subject key identifier    Key info     Expiration   KeyFile/Tag           Host names (SANs)
-------------------------------------------------------------------------------------------------------------
8E1D 5236 BBC1 3A1C ...   NIST P-384    89.7 days   keyfile.kyr           zerossl2.iris.csi-domino.com
0BAE 8710 D30C 1631 ...   RSA 4096      54.3 days                         www.domino-lab.net mail.domino-lab.net
7272 955E 213C D4DD ...   NIST P-384    76.5 days                         *.digitalocean.domino-lab.net
-------------------------------------------------------------------------------------------------------------
3 TLS Credentials


Subject key identifier    Key info     OCSP  KeyFile/Tag           Host names (SANs)
-------------------------------------------------------------------------------------------------------------
8E1D 5236 BBC1 3A1C ...   NIST P-384   OK    keyfile.kyr           zerossl2.iris.csi-domino.com
0BAE 8710 D30C 1631 ...   RSA 4096     OK                          www.domino-lab.net mail.domino-lab.net
7272 955E 213C D4DD ...   NIST P-384   OK                          *.digitalocean.domino-lab.net
-------------------------------------------------------------------------------------------------------------
3 TLS Credentials
```

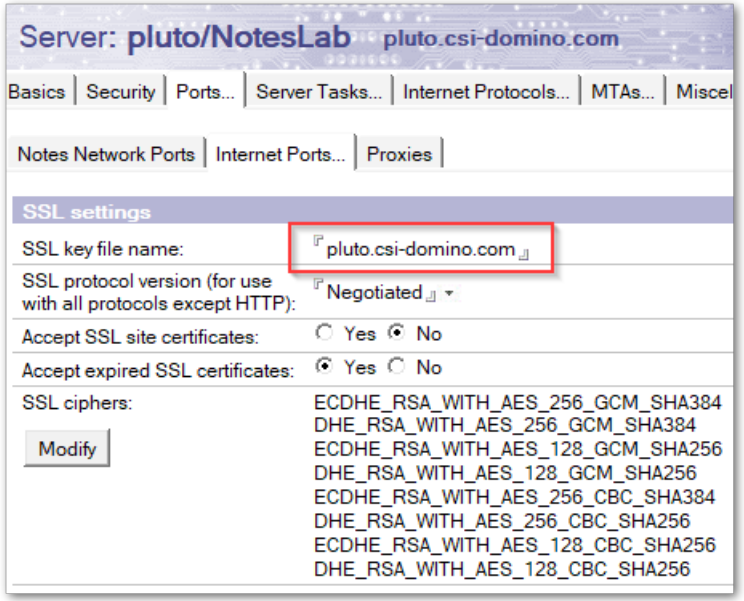
Domino 12+ TLS Cache

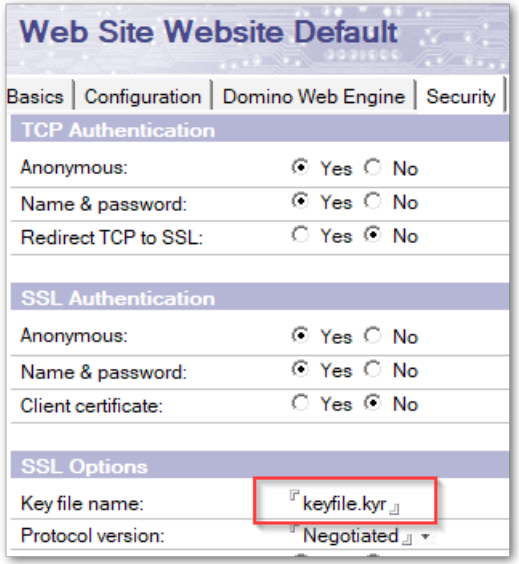
- *.kyr files have been managed by the KYR-Cache reading *.kyr files from disk
- **New TLS Cache reads TLS Credentials directly from certstore.nsf**
- TLS Cache sits in the SSL layer below internet protocols processes (e.g. HTTP/SMTP)
- Support for RSA and ECDSA keys in parallel
- **Support for wildcard certificate lookups**
- **Automatic on the fly certificate reload**
  - when added or updated
- Also manages trusted roots & OCSP cache



Keyfile name field is still very important!

- The **keyfile field in server document** and internet site is still triggering SSL
- Defines the default TLS Credential for the server
- Also used when server acts like a client (e.g. outgoing secure SMTP)
- **Best practice:**
  - Specify Domino server's **host name**
  - Works wildcard certs and RSA + ECDSA certs
  - Not only for HTTP -- Important for SMTP, LDAP, POP3, IMAP





HCLSoftware

HCLSoftware

ACME Support

Let's Encrypt

Automatic Certificate Management Environment

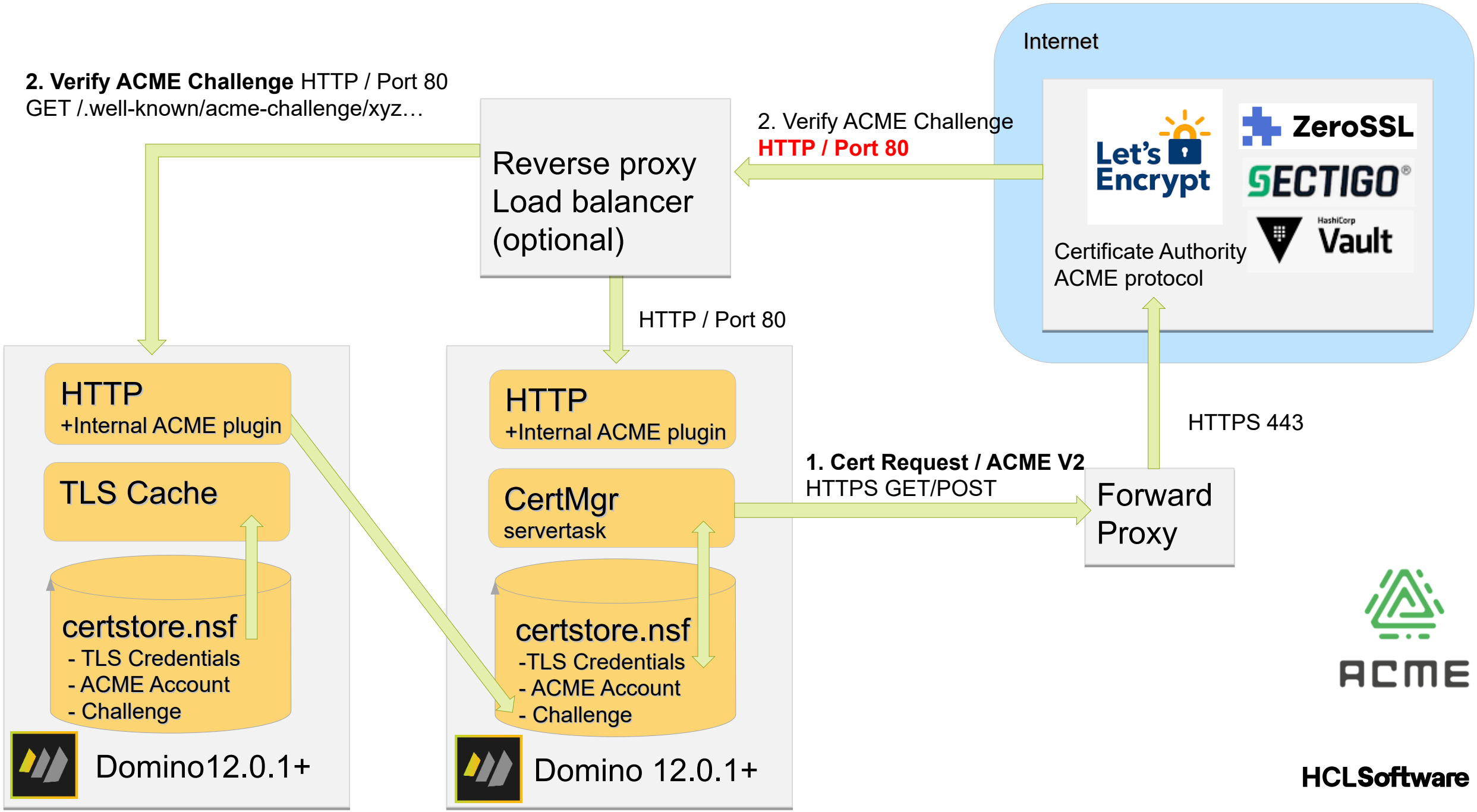
- **ACME is designed for automatic certificate management**
  - RFC 8555 - Automatic Certificate Management
- **The most well known provider is Let's Encrypt**
  - There are additional free and commercial providers
  - I have tested all I could get an account for
  - If you have tested for example with **DigiCert**, **Global Sign** let me know
    - There are no free accounts available and certificates are pretty expensive
- Private CA providers supporting ACME
  - **Small Step CA** is already around for a while
  - **HashiCorp Vault** now also supports ACME
    - CA/Sub-CA, ACME support, SSH Key signing and more, HSM support

ACME HTTP-01 Challenges

- **How it works**
  - 1. ACME server provides a challenge to ACME client
  - 2. ACME server will ask via in-bound HTTP port 80 for the “secret” at a well-known URL

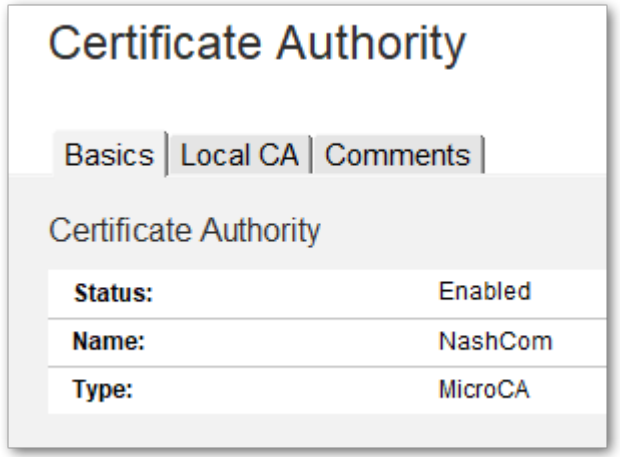
- If server is configured to only allow authenticated connection configure public URL
  - Notes.ini: **HTTPPUBLICURLS=/.well-known/acme-challenge/*:/redir.nsf/*:/MFASetup***
- **Inbound HTTP port 80 is required – Not only for first certificate request!**
- If the server is not reachable by the ACME server (e.g. Let's Encrypt), the challenge fails !!!
  - Tip: Inbound connection can be a proxy connection

- **Troubleshooting document**
  - https://opensource.hcltechsw.com/domino-cert-manager/troubleshooting_acme_challenges/


2. Verify ACME Challenge HTTP / Port 80
GET /.well-known/acme-challenge/xyz…
Reverse proxy
Load balancer
(optional)
2. Verify ACME Challenge
HTTP / Port 80
Internet
Let's Encrypt
ZeroSSL
SECTIGO
HashiCorp Vault
Certificate Authority
ACME protocol
HTTP / Port 80
HTTP
+Internal ACME plugin
TLS Cache
certstore.nsf
- TLS Credentials
- ACME Account
- Challenge
Domino12.0.1+
HTTP
+Internal ACME plugin
CertMgr
servertask
certstore.nsf
-TLS Credentials
- ACME Account
- Challenge
Domino 12.0.1+
1. Cert Request / ACME V2
HTTPS GET/POST
Forward
Proxy
HTTPS 443
ACME
HCLSoftware

HCLSoftware

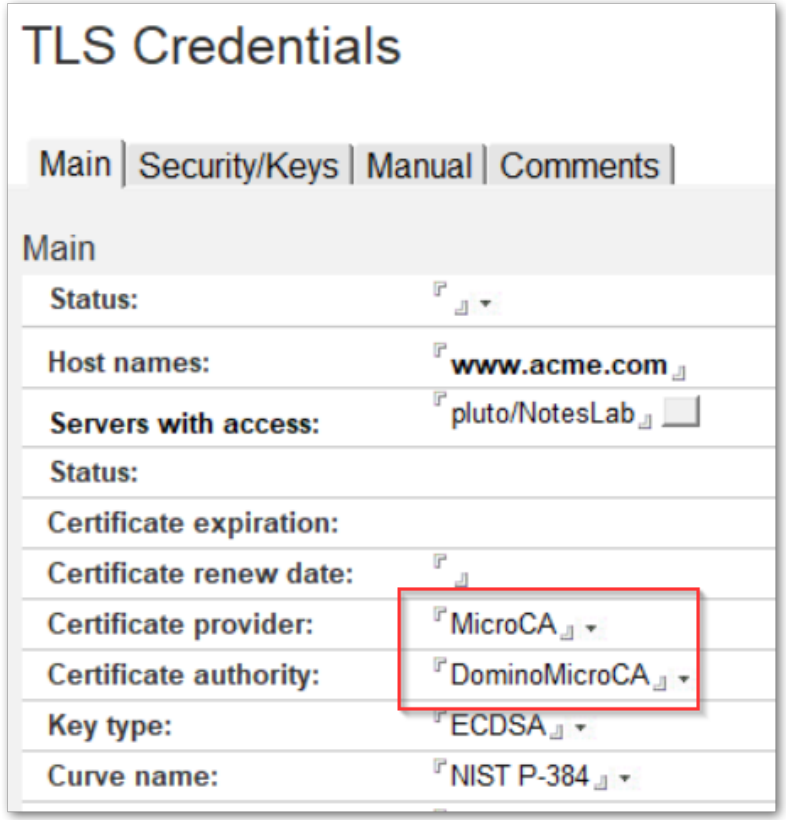
Domino Micro CA

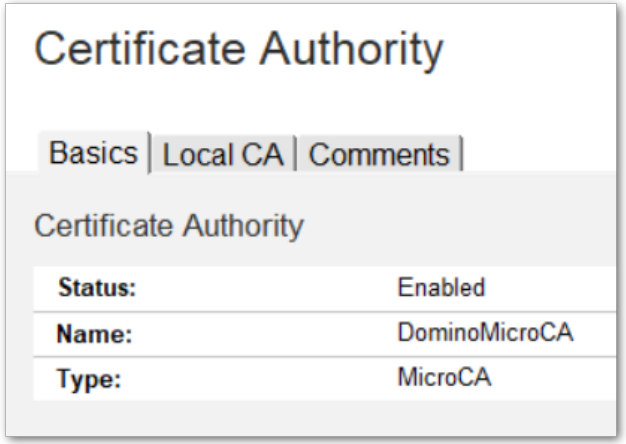


Simple internal Micro CA

- **Domino 12.0.1 introduced a simple "Micro CA" managed by CertMgr**
  - Available via **One-touch Setup** for first server in domain
  - Or from certstore.nsf to issue a certificate from the **local CA**
  - Not a full CA – But enhanced from release to release

- A very good option for internal certs
  - For examples to be used behind load-balancers
- Improved in every release – including 14.5.1
- **Can be also used as a internal CA**
  - Recommendation update to 14.5.1
  - With 14.5.1 use shorter certificate expiration

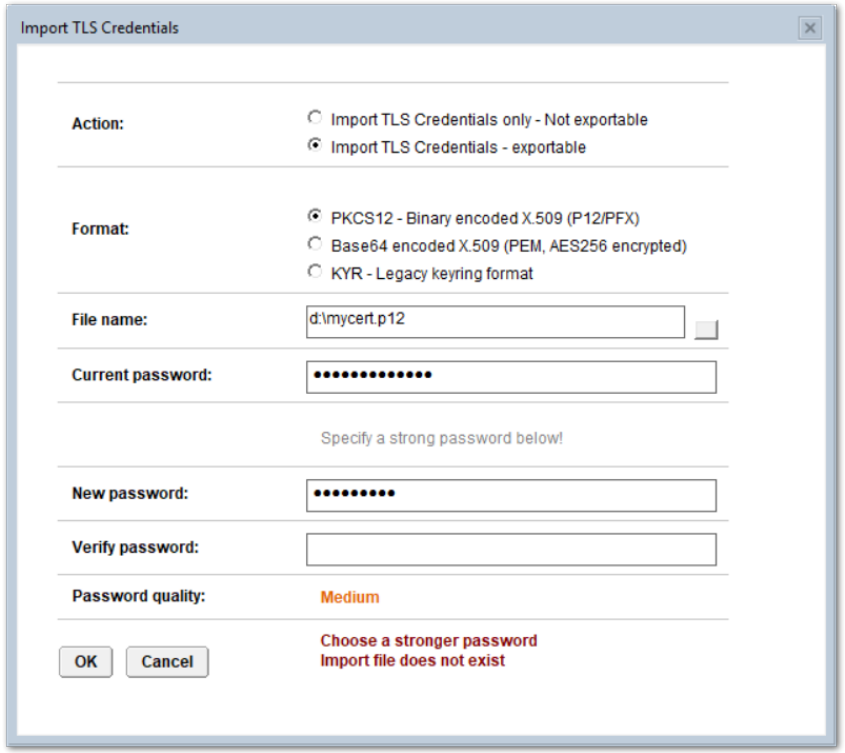




HCLSoftware

HCLSoftware

Export & Import

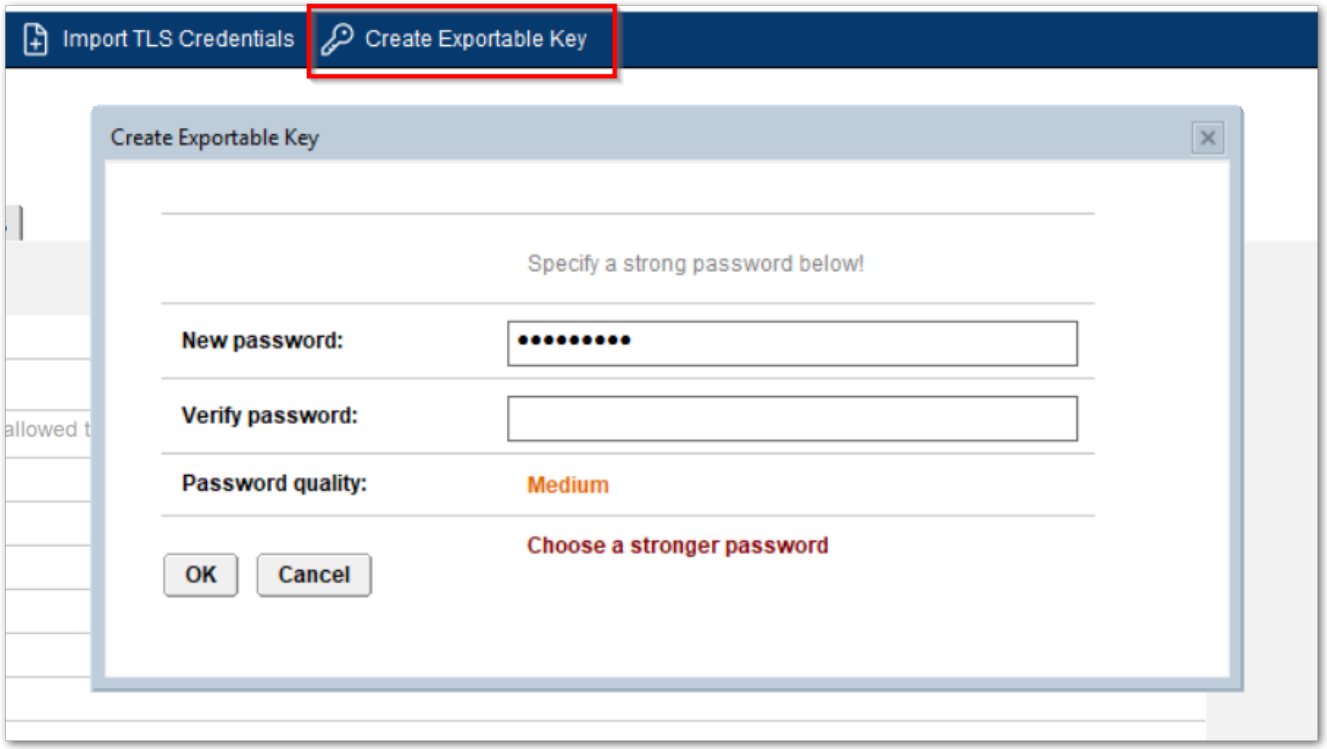


Concept of “exportable keys”

- The private key is **always** encrypted for servers only and can be only decrypted by
  - **CertMgr server**
  - **All servers listed in “Servers with access”**
- Beginning with Domino 12.0.1 private keys can be created or imported “**exportable**”
  - a.) Either create an “exportable private” key for ACME / manual / MicroCA certificate flows
  - b.) Or import x.509 certificates marking them exportable
- Exportable keys are protected with a password/passphrase
  - Stored as **AES 256 encrypted PEM** in a visible field in the form
- Export functions use the Exportable key field to create files containing the complete TLS credentials information (**key + cert + chain + root**)

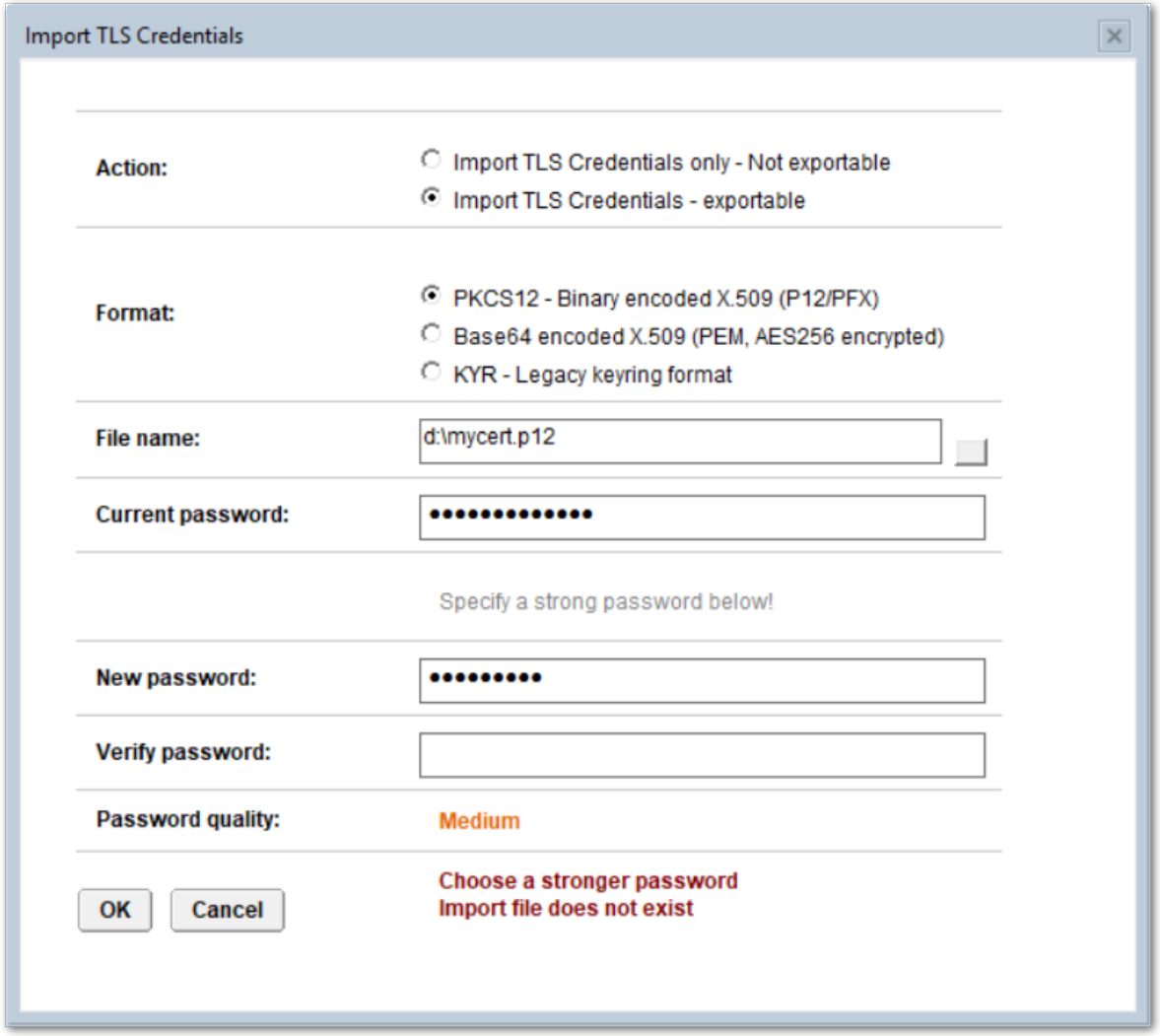
Create Exportable Key

- Create key and specify a secure password
  - Export requires password with sufficient entropy
- Create based on key type size/curve
- Created key can be used for all flows
  - ACME
  - Manual
  - MicroCA



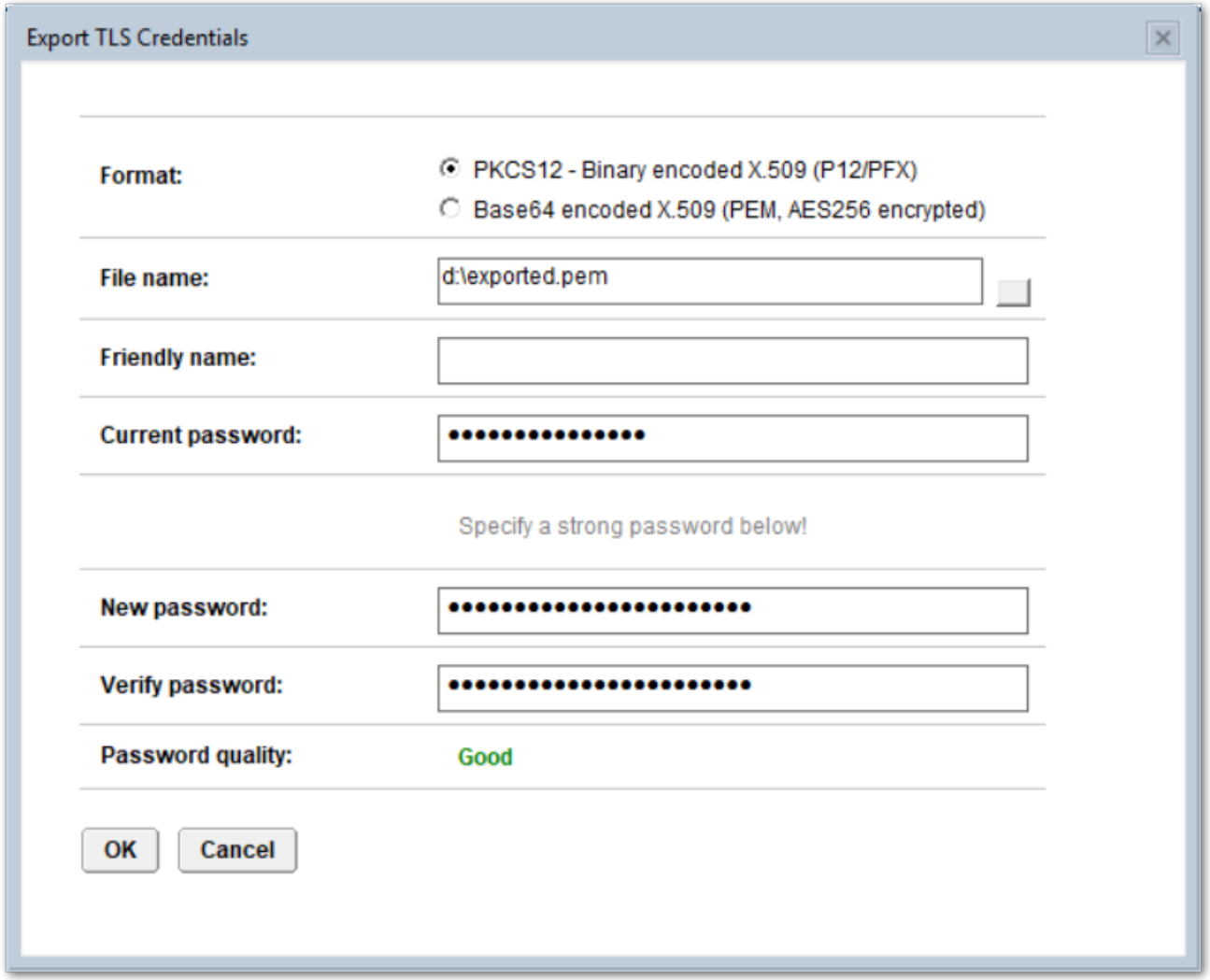
Import existing X.509 Certificate and Key

- Supports
  - **PEM (.pem)**
  - **PKCS12 (.p12/.pfx)**
  - **Keyring files (.kyr)**
- Default option: Import only
- Import and mark private key exportable
  - Requires password for exportable key
- Supports password protected files
- Sorting and filtering certificate data
- Auto complete chains



Export TLS Credentials

- Supports
  - **PEM (.pem)**
  - **PKCS12 (.p12/.pfx)**
- Export action is only shown when exportable
- Only password protected / encrypted export with current encryption standards!
  - **PEM: AES 256**
  - **PKCS12: AES 256 / PBES2**
    - See: https://datatracker.ietf.org/doc/html/rfc8018
    - Not all external tools might be able to read it



HCLSoftware

Implementation

- **Single sub-form** for **export**, **import** and **create exportable key**
- Designed to use "**Enter**" to validate the form (OK button) for convenience
- Validation is built into the form
  - All errors are shown on the bottom of the dialog box at once
- Script lib encapsulating back-end functionality
  - Leverages a 12.0.1+ C-API call in nnotes.dll/libnotes.so
  - **Export/import actions are only shown in 12.0.1+ clients**
- Central call used for all export and import operations
  - **Load certmgr -importpem / -importkyr**
  - One-touch Domino setup: Import existing ***.pem / *.p12 / *.kyr** files during setup!
    - Note: Only available for first server in the domain

Use cases for import & export

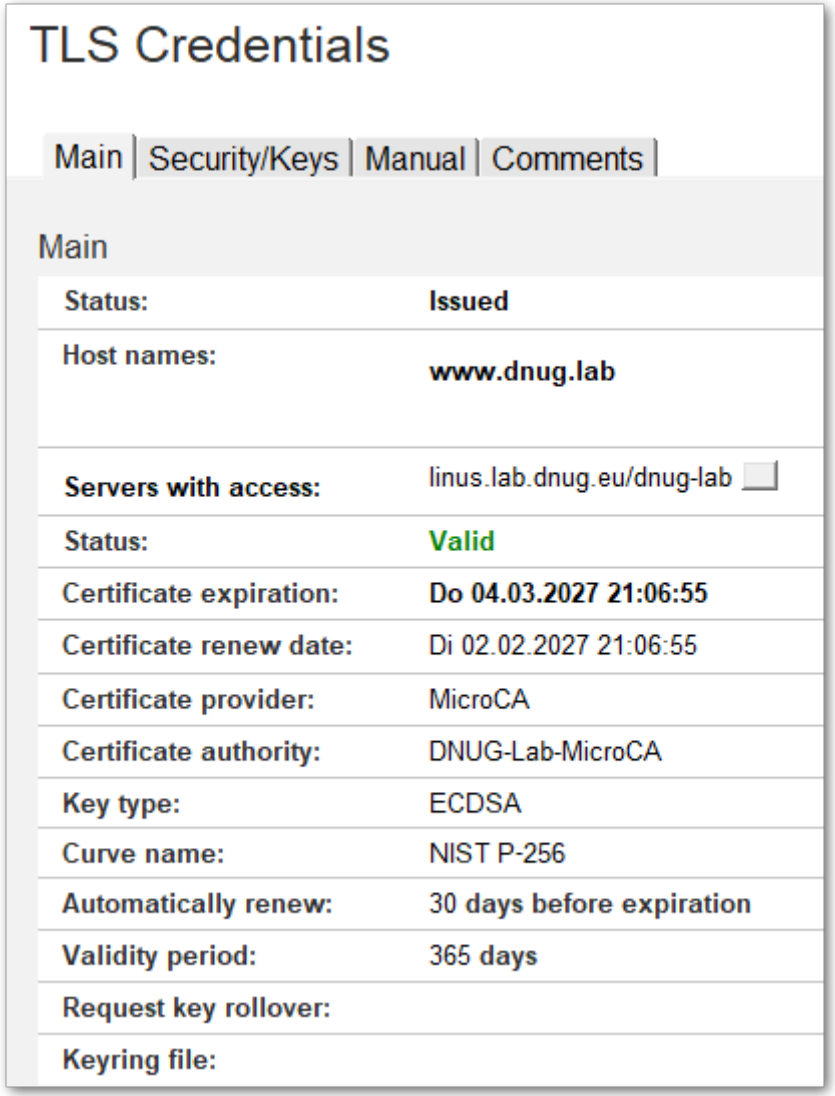
- **Import existing X.509 certificates**
  - Use existing PEM (**.pem**), PKCS12 (**.p12 / .pfx**) and keyring-files (**.kyr**)
    - Like wild card certificates already available in your organization
    - Support for **password protected PEM** and **PKCS12** with current crypto standards
- **Export CertMgr TLS Credentials to be used on other environments**
  - **Export** to encrypted **PEM** and **PKCS12** files
    - Exported keys are required to be password protected with a “secure” password
    - Useful to export for Sametime, SafeLinx load-balancers & Co
- Most automation functionality can be triggered by writing the right fields
  - Extended automation requires the LS2CAPI Call (for example creating private keys)
  - Open source Script Lib available for export & import from Nash!Com on request

HCLSoftware

What's new in 14.5.1

MicroCA Improvements

- The MicroCA was started very simple just for the AppDevPack
- Meanwhile it got more functionality from release to release
- Validity period can be specified for each TLS Credentials doc
  - Default 365, can be shorter or longer depending on use case
- **Modern certificate extensions**
  - Subject Key Identifier (SKI)
  - Authorized Key Identifier
- Not displayed in TLS Credentials form but part of certificate.
  - Can be displayed with OpenSSL and other tools…
  - See example on next slide



Manual Flow Improvements

- **Specify emailAddress**
  - Some companies require an e-mail address specified for internal CA certificate requests (CSR)
  - An email address isn't what a web server certificate should use
  - But not all attributes need to be added by the CA from CSR
- **Tolerate certificates with IP and email SANs**
  - Domino only supports DNS server SAN extensions
  - ACME and Micro CA flow only generates DNS SAN certs
  - IP and email SANs could still be imported via manual flow
  - Starting with 14.5.1 only the DNS SAN extension is read to allow to use certs with IP address SANs
    - In earlier versions the import fails

Cert Details

- Subject Key Identifier (SKI)
  - is usually a SHA1 of subject public key but could be also the first 20 bytes of a SHA256 hash
- It is calculated and added as an extension
- Authorized Key Identifier
  - Is the SKI of the issuing CA cert

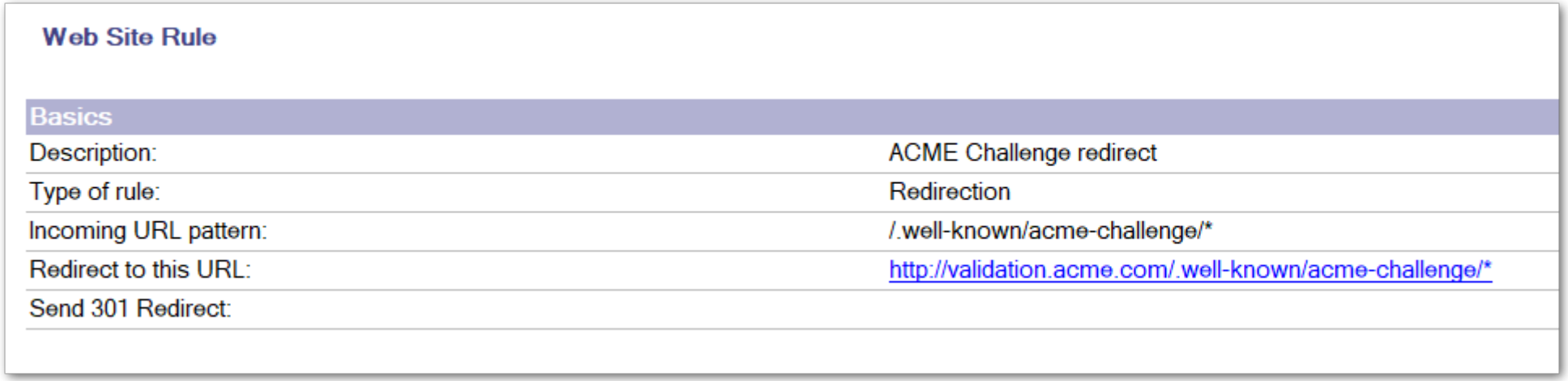
```
#0
Subject     : /O=dnug-lab/CN=www.dnug.lab
SAN (DNS)   : www.dnug.lab
Issuer      : /O=DNUG-Lab-MicroCA/CN=CA
Not Before  : 2026.03.02 20:06:55
Not After   : 2027.03.04 20:06:55 (expires in 365.0 days)

Serial      : 7B70DDA44EC22127C171BEFDB1CCB157
Sign Alg    : ecdsa-with-SHA256
KeyUsage    : DigitalSignature, NonReputation, DataEncipherment, KeyAgreement, CrlSign, EncipherOnly, DecipherOnly
Extensions  : ExtKeyUsage
ExtKeyUsage: TLS Web Client Authentication, TLS Web Server Authentication
Key         : ECDSA NIST P-256
ASN1 OID    : prime256v1

AuthKeyId   : 6B:6B:E7:CB:12:E6:29:4C:C2:14:4F:77:69:A4:EE:02:CA:52:10:6E
SubjKeyId   : 73:56:DD:08:AD:7E:05:CB:B6:81:A7:20:B3:8D:4B:66:3E:6E:B1:F2
SKI-SHA1    : 73:56:DD:08:AD:7E:05:CB:B6:81:A7:20:B3:8D:4B:66:3E:6E:B1:F2
SKI-SHA256  : 98:8D:C8:9B:C5:D1:26:A4:40:D6:EC:FA:D5:30:40:FE:90:95:9C:7D:34:9E:59:77:D5:58:C6:6D:9D:6E:54:56

MD5         : 8C:F3:ED:6D:59:AF:76:DE:12:FD:25:89:C0:EA:3A:91
SHA1        : 51:BA:1D:BA:9C:B5:2E:1B:E8:6D:78:2E:B3:EA:CF:BE:06:3B:C0:01
SHA256      : 11:D9:A6:F0:17:F9:70:8F:2E:16:64:EC:66:45:4F:AB:80:5F:8A:FF:9B:F8:89:41:12:EC:1D:67:BD:1E:01:2C

#1
Root        : /O=DNUG-Lab-MicroCA/CN=CA
Not Before  : 2025.11.25 20:40:21
Not After   : 2035.11.27 20:40:21 (expires in 9.7 years)

Serial      : 01E87A804EC6D9C8E6CFFC866BE9F7C7
Sign Alg    : ecdsa-with-SHA256
Extensions  : BasicConstraints, CA, SelfSigned, KeyUsage
Key         : ECDSA NIST P-384
ASN1 OID    : secp384r1

SubjKeyId   : 6B:6B:E7:CB:12:E6:29:4C:C2:14:4F:77:69:A4:EE:02:CA:52:10:6E
SKI-SHA1    : 6B:6B:E7:CB:12:E6:29:4C:C2:14:4F:77:69:A4:EE:02:CA:52:10:6E
SKI-SHA256  : 51:CD:61:8E:7C:0C:22:33:A1:C6:2C:6C:63:0F:DD:E2:64:A4:AD:EF:88:C6:1F:04:95:90:6E:57:74:94:19:F2

MD5         : 16:5E:F7:FC:DD:FF:84:08:AF:EB:0E:82:CE:F4:B6:C3
SHA1        : 9E:E6:49:35:4F:5E:2D:8B:83:33:5E:C6:34:51:1C:EE:A4:95:85:BD
SHA256      : 02:7F:67:F5:90:1D:2D:05:9E:C8:F6:E6:B8:88:24:C9:06:CB:FA:5F:65:5D:71:EA:CC:C3:7F:88:50:CB:A5:79
```

HCLSoftware

Extending CertMgr

Redirect ACME HTTP-01

- ACME HTTP-01 challenges always need to hit HTTP **Port 80** first
  on the server pointing to the DNS name first
- You can redirect all challenges to a single server reachable by ACME providers
  - Redirect targets can be any server on HTTPS/HTTP on port **80/443**
    - Certificate on redirect https:// target server is not checked – Just needs to respond via HTTPS
  - Simple redirect rule on server where the DNS name points to
  - Also works with other type of servers (e.g. NGINX)



Web Site Rule

Basics	
Description:	ACME Challenge redirect
Type of rule:	Redirection
Incoming URL pattern:	/.well-known/acme-challenge/*
Redirect to this URL:	http://validation.acme.com/.well-known/acme-challenge/*
Send 301 Redirect:	

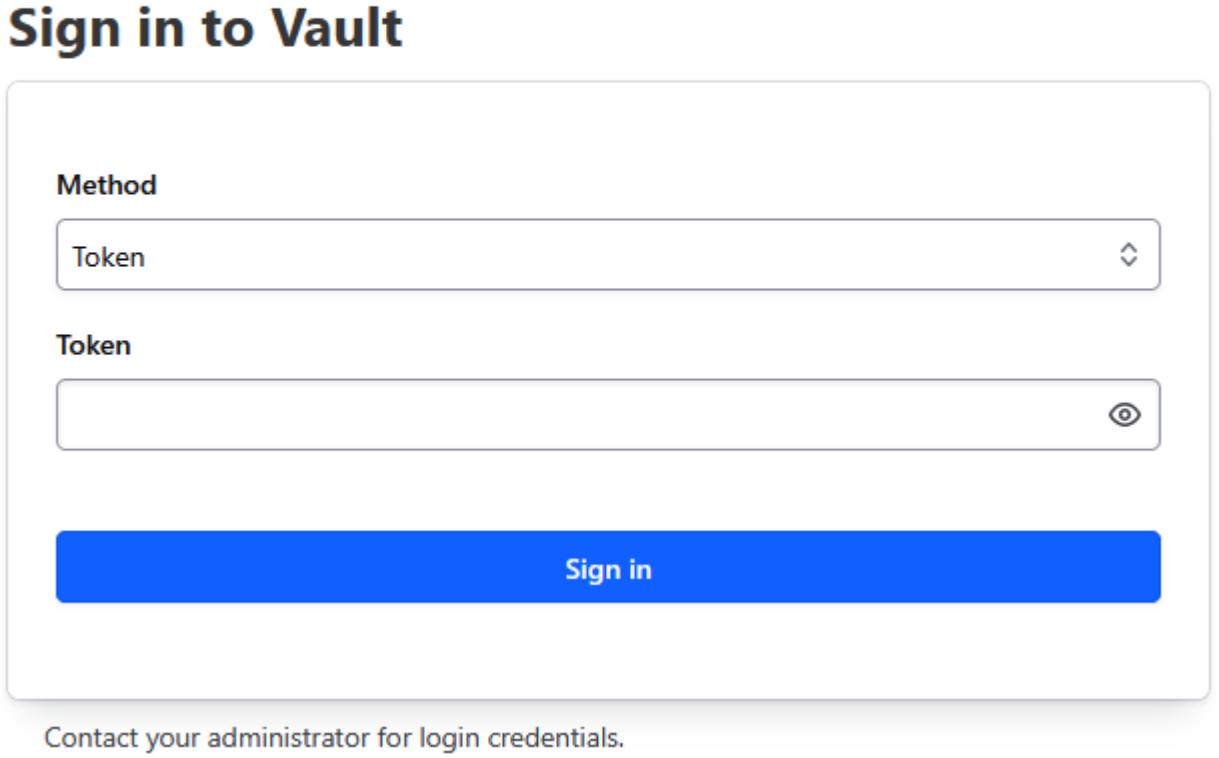
Integration - Load Balancers & Proxies

- Redirect configurations for Load Balancers
  - Redirect HTTP to HTTPS
  - Redirect ACME challenge to another server via rules
- Transfer exportable keys
- Retrieve certificate data matching private key remotely from CertMgr server
- Config + Script
  - https://github.com/HCL-TECH-SOFTWARE/domino-cert-manager/tree/main/examples/nginx

```
# Port 80 is redirected to 443. Only ACME challenges are redirected to CertMgr server.

server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;

    location /.well-known/acme-challenge/ {
        return 301 https://certmgr.acme.com$request_uri;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}
```

HashiCorp Vault

- **Cloud native, API first CA**
  - Very flexible for integration
  - Can act as a Sub-CA
- Supports HSM hardware
- **Supports ACME CA**
  - https://www.hashicorp.com
- Good choice for internal deployments
  - Can act as a Sub CA for an existing internal CA
  - OIDC, JWT, GitHub, Okta, Radius, .. support
- Very good choice for an internal CA with integration
- Automated Lab deployment → https://github.com/nashcom/nsh-vault-deploy



HashiCorp
an IBM Company

HCLSoftware

Questions & Answers

- Thanks for your interest!
  Open questions? Feedback?

- Additional Resources
  - Bonus Material appended to this slide deck
  - https://opensource.hcltechsw.com/domino-cert-manager
  - https://github.com/nashcom/srvguard/blob/main/docs/certificate-lifetime-reduction.md

- Ask questions via GitHub Issues
  - https://github.com/HCL-TECH-SOFTWARE/domino-cert-manager

HCL Domino

Nash!Com
Communication Systems

HCLSoftware

Additional Material

RSA & ECDSA keys

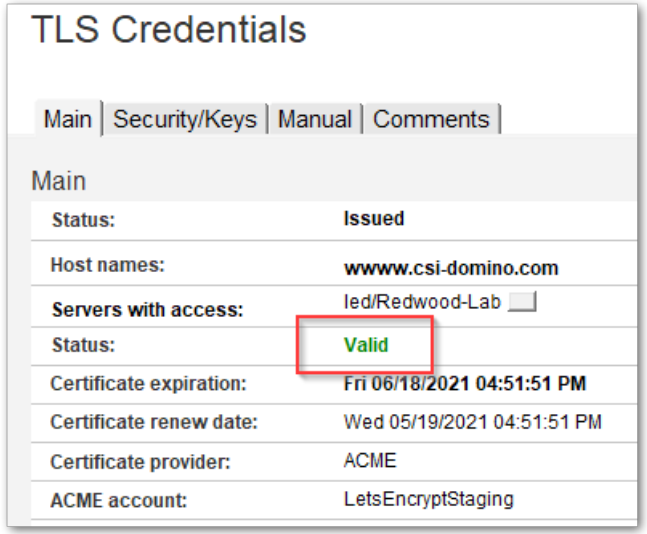
- **RSA** private/public key encryption is the most widely used standard
  - Notes was one of the first application using it very early - long time ago..
- **ECDSA** keys (ECC) are based on elliptic curve cryptography
  - Much shorter keys for comparable key strength
  - **Require much less CPU**
  - Different types of curves
    - Numbers should not be referred to as “bits”
    - Refer key type with a “curve” : “ECDSA NIST-P256”
  - Notes/Domino 12.0.2+ fully supports ECDSA
- Tip: Use **ECDSA NIST-P256** instead of **RSA** keys

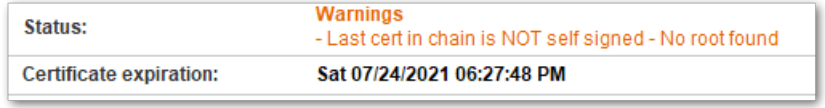
```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIPvL5AK+h6R6GnpuePXtmhkTdqCrXPR/v0yXheSEHkMYoAoGCCqGSM49
AwEHoUQDQgAE3feNYjJOmK5CD+nh75mXeYQjfpWYJzh5N5rq6NjKvlwnR/M0CqdE
kHafeyVVzkixsn3R7oTgGJYkq3hzA5FPRA==
-----END EC PRIVATE KEY-----
```

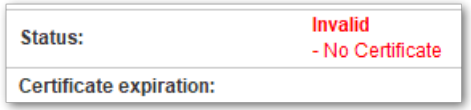
```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQCZwxbw5yRayMERfZ5iwL+ECFguVI3H67hWuHeu3sWUSj89j/hf
u03hsFJJ+09GxHEdQrdgxgcL9g7dSZtAA40b+/uLus5UtCogyGYMbWMdW3dINEwr
ii3NN1jqWAUn8Kego3fWESRAGvrVGqeQ0Fryi46KVemWZs3nQ0hPedqcTQIDAQAB
AoGAOHadfAVFIllKHKQmf1kMeu2dhBXkop96582BlaOUpmnFY1P0/yKG7POUpDo2
2GPWrcoMEOtnNws6ViRrcExdinFsDnoO6RI1Z4DjBsKr79sGTzx/FV5yrDKeoKhu
nmTY5iM1NbXkDhs3I8Z4aamLTiNf3HAM3YKH5kRoRcnT87kCQQDMiIpHm+GrzqvZ
Co1wUeB1Y2cS7v/B1kMbNfBCFrhmaXM2wnQsi9lVYbHK2nWw634pEKAQgIT/zbWn
iBr9T7bDAkEAwHQBtD2gvYdUGtZnjAFnX9TspxgX1O+rPgo4fqdedkX5jjO5TKyc
0VQiFPlrhVTdThFqXikT9jU5M0OubsfPrwJBAKVM/KTfIMXf5BcnZiQOqB93VMiC
jAU2i0mkkCZ5glKSiuEIyydtZxqQOEa/xDpWOMIrg5GSrOqocFWTtGxhTk0CQQCP
PUiPaFcc+X5lwzKqh9jzAL1q2InUWNXxhRXrXpL2ILrPVjwSj17ghmgfiEy4niNh
GaE7mANdetuqBjPsDBfNAkAwhvswm3q+b2JF1dPpY1SAjAjkRxmfVhPvrTPSnF+Z
tcLhcmjSmOvpGjn8q218fXq7fevlnpn8l0oRjHvonsap
-----END RSA PRIVATE KEY-----
```

Certificate Health Check & Inspecting Certificate Chains

- **All certificate operations check if the certificate is valid**
  - Status: Valid, Warnings, Errors
  - Detailed warning and error messages
- Most common warning:
- **“Last cert in chain NOT self signed – No root found”**
  - Not an error - Just means that there is no trusted root
  - Trusted roots can be imported separately and are added to the chain if present
- Updates CertMgr statistics to reflect the current health status








```
> show stat certmgr.*
CertMgr.CertStatus.Green = 3
CertMgr.CertStatus.Red = 1
CertMgr.CertStatus.Yellow = 2
CertMgr.CertStatus = Red
4 statistics found
```

Support for Elliptic Curves – “ECDSA Keys”

- **ECDSA** is the more modern, more secure standard with less overhead
  - **NIST P-256** ECDSA key → **3072** bit RSA key or a **128** bit AES key
  - **NIST P-384** ECDSA key → **7680** bit RSA key or a **192** bit AES key
  - **NIST P-521** ECDSA key → **15360** bit RSA key or a **256** bit AES key
- Fully supported in the Domino V12 TLS/SSL stack
  - Support for RSA and ECDSA key types in parallel – even for the same host!

- With ECDSA the following ciphers are automatically used instead of the cipher config
  - **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xC02B)**
  - **TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xC02C)**

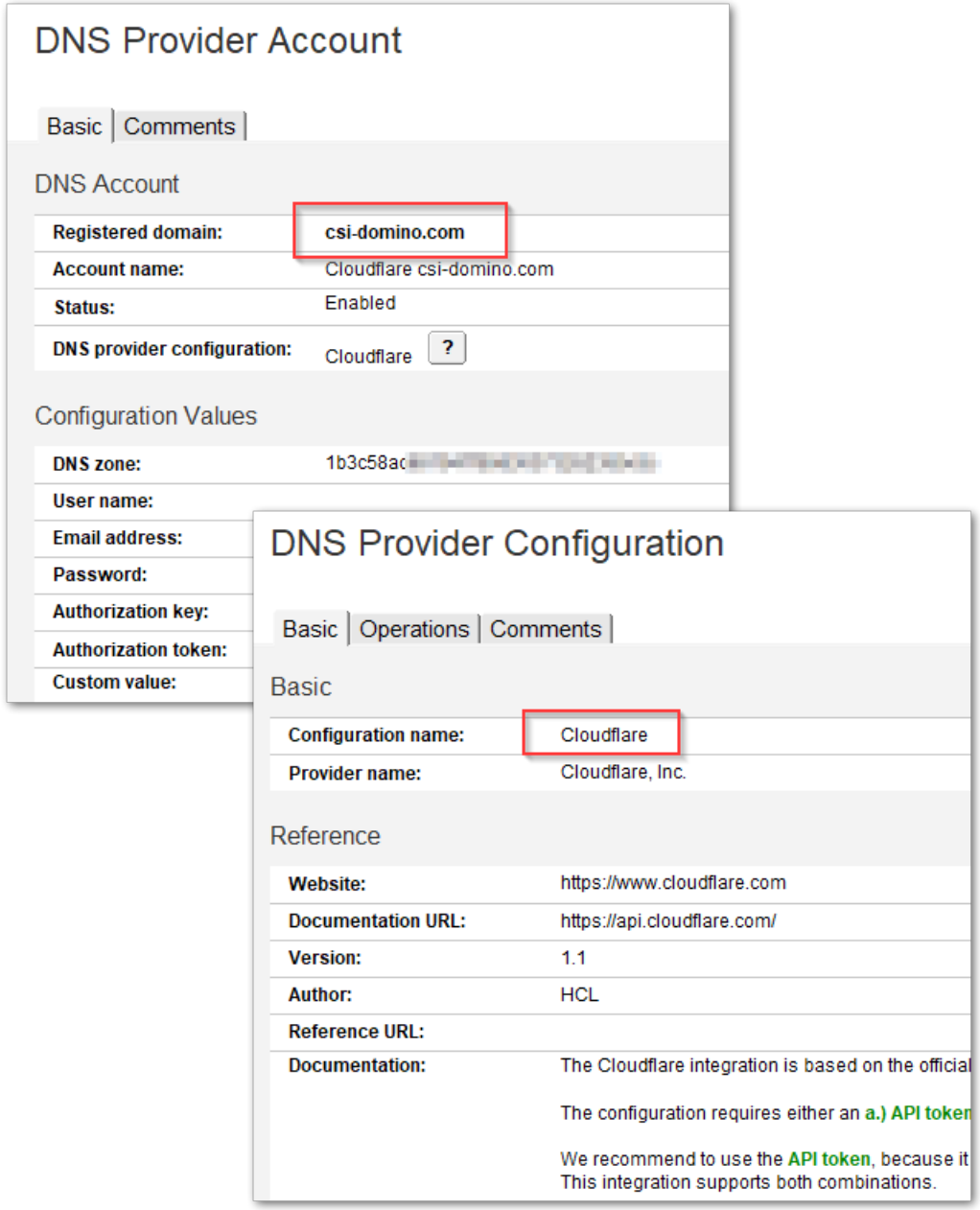
- Background Information ECDSA
  - https://blog.cloudflare.com/ecdsa-the-digital-signature-algorithm-of-a-better-internet/

Certificates

- **Keys are signed using the private key** of a Certificate Authority (CA)
  - **Validated against the root certificate**
- Usually there is a “certificate chain” with multiple Sub CAs creating intermediate certificates
- The root certificate is stored in a **local trust store** (e.g. installed in your web browser)
  - Servers can send the root certificate for completeness. But it is not required
  - It’s still good to have the full chain locally to verify your certificate chain is complete!
- Leaf certificate and intermediate certificates are sent by the server
  - **Used to verify the certificate chain from leaf certificate to root certificate from your local trust store**
  - **Verification of a certificate always needs the full chain!**
- Let’s have a look in detail..

DNS Provider Interface

- Triggered by configured “registered domain”
  - Choose the right provider
  - Specify provider specific information
- Build-in support to integrate DNS providers
  - Easy to integrate REST interface (@Formulas)
    - Recommended interface!
    - Works for most providers
    - “Low code approach”
- Notes Agent
- Command line Integration



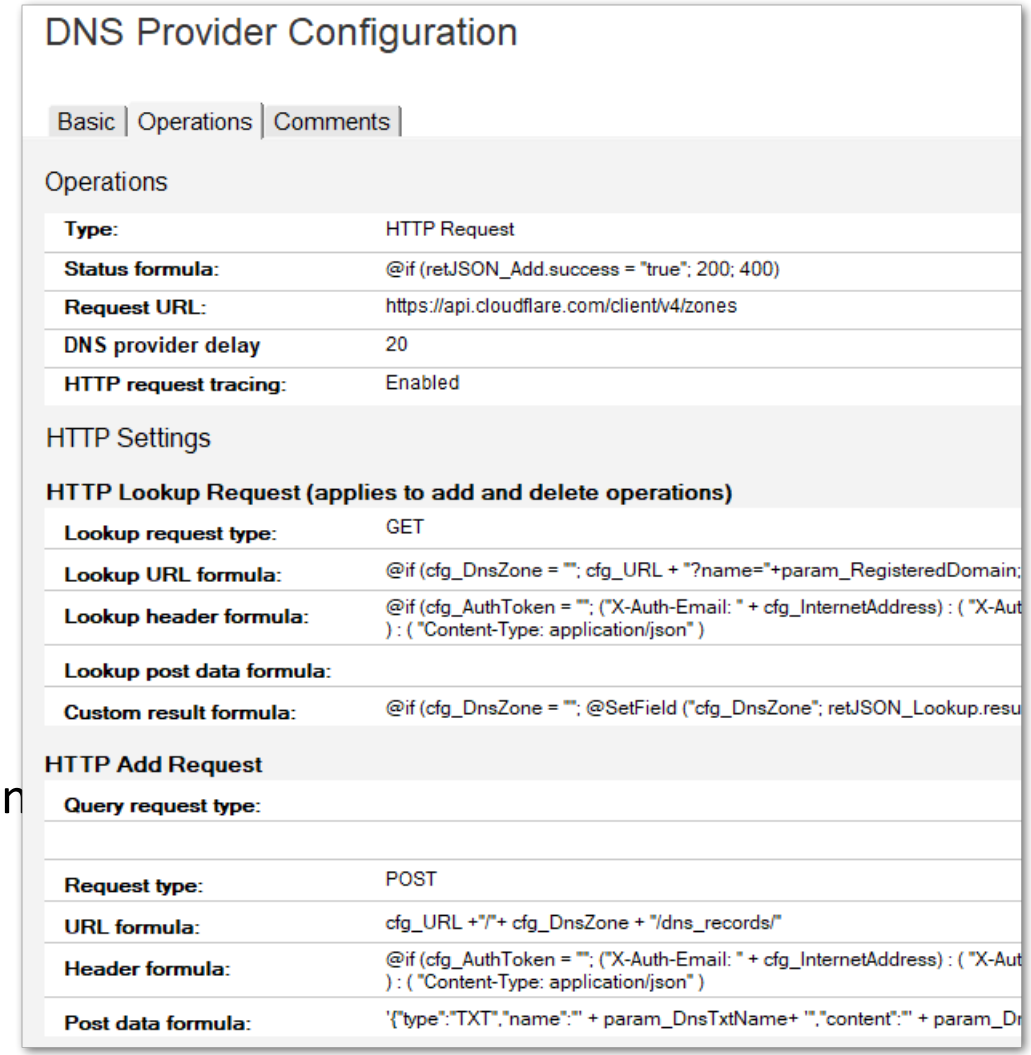
ACME DNS-01 Challenges & Wild Card Certificates

- Allows to request certificates without inbound internet connection
- ACME Challenge is stored in DNS TXT record

- Supports wild card certificates! e. g. *.acme.com
- Requires DNS provider API with outgoing HTTPS connection to DNS provider
  - No inbound connection needed
  - Can leverage outgoing proxy connections
- CertMgr server can request certificates for any server in the DNS domain

DNS Provider Interface – Import DXL

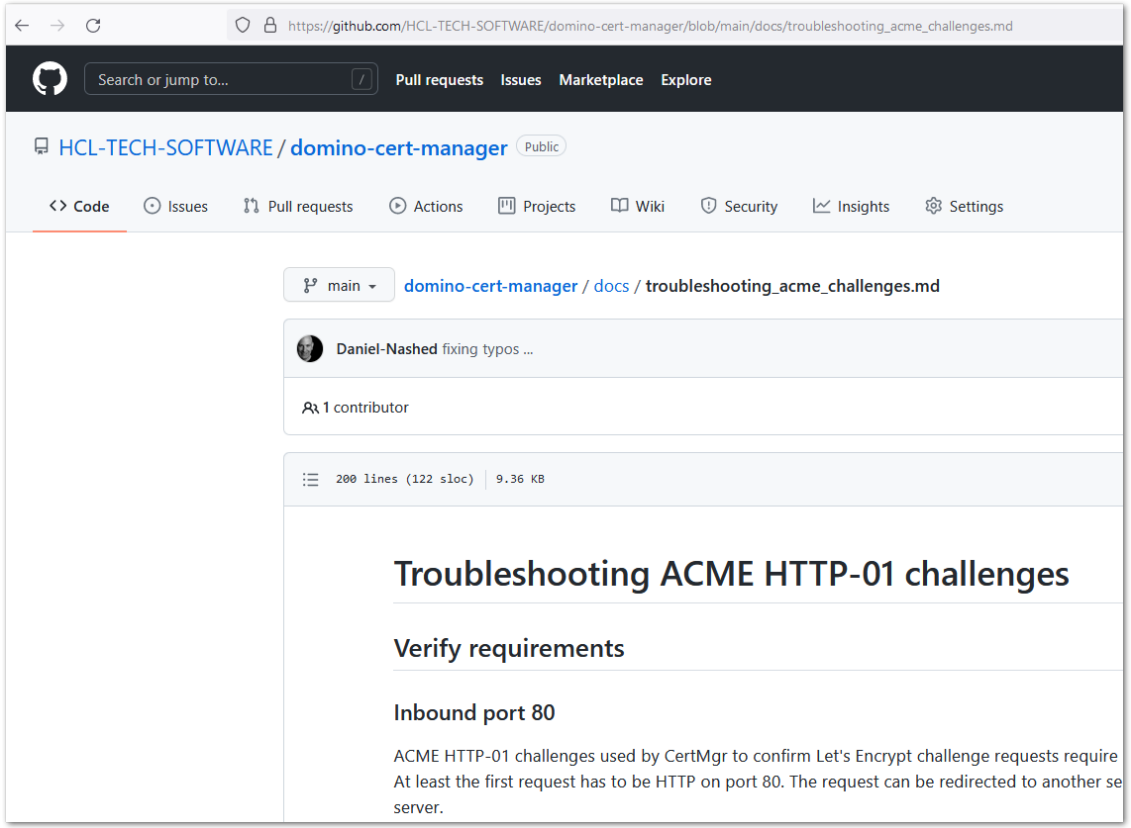
- DNS TXT API DXL files are not included in certstore.ntf
- Provider interfaces can be imported via DXL files
  - Database action: Import DXL
- REST interface is based on @formulas
  - Low code approach
  - Can parse JSON responses
  - Helper buttons for inserting fields & testing formulas
  - Trace results useful for troubleshooting and developmen



HCLSoftware

HCL GitHub Repository “domino-cert-manager”

- ACME DNS-01 challenge integration
  - Scripts for integration with DNS TXT API
  - Questions for DNS TXT API integrations should be issues on GitHub instead of HCL support tickets
- Troubleshooting information for ACME HTTP-01
- Information about ACME providers etc.
- Central resource for ACME integration



HCLSoftware

Domino CertMgr Commands

- **Tell certmgr show certs**
  - Shows the currently loaded TLS Credentials for this server
  - Each process has it’s own cache
    CertMgr server task is used to show certificate information
- Tell command only works on CertMgr server – Not on certmgr task in client mode
  But option is also available on all servers (Windows/Linux) via
  - **load certmgr -showcerts**
- Tell certmgr show ocsp
  - Checks the OCSP status for all configured TLS Credentials
  - Requires OCSP to be enabled
  - Also available via load certmgr -showocsp

CertMgr Command Examples

```
Subject key identifier    Key info     Expiration   KeyFile/Tag           Host names (SANs)
-------------------------------------------------------------------------------------------------------
8E1D 5236 BBC1 3A1C ...   NIST P-384    89.7 days   keyfile.kyr           zerossl2.iris.csi-domino.com
0BAE 8710 D30C 1631 ...   RSA 4096      54.3 days                         www.domino-lab.net mail.domino-lab.net
7272 955E 213C D4DD ...   NIST P-384    76.5 days                         *.digitalocean.domino-lab.net
-------------------------------------------------------------------------------------------------------
3 TLS Credentials


Subject key identifier    Key info     OCSP  KeyFile/Tag           Host names (SANs)
-------------------------------------------------------------------------------------------------------
8E1D 5236 BBC1 3A1C ...   NIST P-384   OK    keyfile.kyr           zerossl2.iris.csi-domino.com
0BAE 8710 D30C 1631 ...   RSA 4096     OK                          www.domino-lab.net mail.domino-lab.net
7272 955E 213C D4DD ...   NIST P-384   OK                          *.digitalocean.domino-lab.net
-------------------------------------------------------------------------------------------------------
3 TLS Credentials
```

TLS Cache Select Criteria

- V12.0.1+
  - Ensure that TLS Cache loads TLS Credentials also in transient or renew error state
  - TLS Credentials with certificate status of “yellow” and “green” with the following status are loaded
    - Issued
    - Pending
    - Renew
    - Waiting
    - Error
    - Expired
    - Update Server List

New algorithms importing/exporting Certificates

- Import now also supports newer AES-CBC with 128/192/256 bit keys.
- For backward compatibility
  - **3DES-CBC, SHA-1** (hmacWithSHA1)
  - **3DES-CBC, SHA-2** (hmacWithSHA256, hmacWithSHA384, and hmacWithSHA512)
- Exporting PEM or PKCS#12 file always use more modern and secure standards
  - **PBES2 with 256 bit AES, 4096 iterations, and HMAC-SHA2**
  - Same default in OpenSSL 3.0 (Tip: 3.0 -legacy switches back to the older weaker encryption if needed)
- For older software like Java 8/11 applications (e.g. Sametime 11)
  you will either need to convert via OpenSSL or lower the export security via notes.ini
  - **PKCS12_EXPORT_LEGACY=1**
  - Downgrades all of the PKCS#12 files exported to use 3DES with SHA-1

Server Name Indication – aka SNI

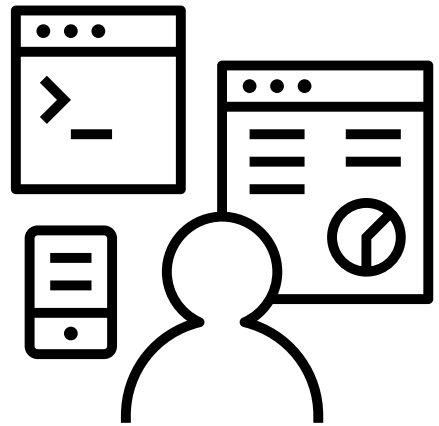
- **Challenge**
  - HTTPS requests are completely encrypted
  - How do you know which key you should use to decrypt the request?
  - Which host do you map the request to?
  - Usually this leads to the requirement to have a separate IP address for each TLS/SSL enabled website
- **Solution**
  - Server Name Indication (SNI) adds requested hostname to TLS/SSL handshake
  - Server can read the SNI name requested and map the request to the right key for decryption
  - Domino 11.0.1 supports SNI Domino 12.0+ supports more flexible SNI flows
- **Restrictions/Limitations**
  - Client must request the website using the SNI extension, Server must support the SNI extension
  - Hostname requested is sent unencrypted and can reveal information about hosts requested

Import TLS Credentials via command-line

- Server side Command Line operations to import existing “SSL Certificates”
  - Early feature introduced in 12.0 before import/export was introduced in 12.0.1
  - Today it is better to use the client side import or the Lotus Script LS2CAPI call from a script lib
- **load certmgr -importkyr keyring.kyr**
  - Import one KeyRing file (*.kyr) to CertStore
  - **load certmgr -importkyr all**
    - Import all KeyRing files referenced from server doc & internet sites to CertStore
    - Intended for migrations and ensures each key is imported only once
- **load certmgr -importpem <file>**
  - Import file containing PEM encoded certificate chain and keys to CertStore
- **load certmgr -ImportRootFromUrl <URL>**
  - Generates trusted root document in **certstore.nsf** from certificate found on server
  - Trusted root is imported as draft and needs to be marked as trusted by admin

HCLSoftware

Troubleshooting



Global Settings

- Mainly used to set defaults for important settings
  - For example: key type, key size, default ACME account and renewal interval

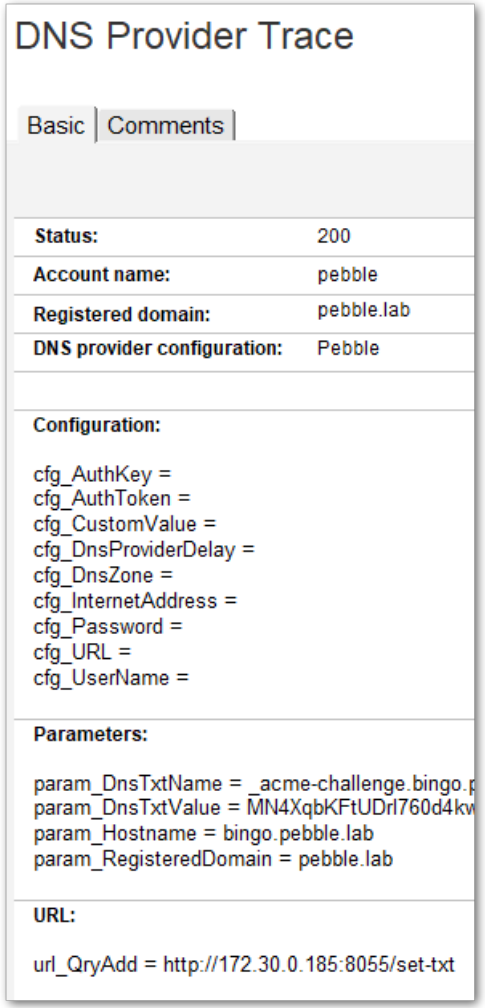
- Admin Server for CertMgr
  - Admin server is also stored in Directory profile “CertMgrServer” to publish in the domain
  - Changing the admin server involves re-encryption of keys
- Migration via command-line option only!
  - Load certmgr **-MIGRATETOSERVER** server-name
  - Re-encrypts all private keys after checking all keys can be read

Additional Notes.ini Parameter

- **CertMgr_ReplicationInterval=n**
  - Default: 120 sec
  - Used for client mode
- **CertMgr_HealthCheckInterval=n**
  - Default 30 minutes
- **CertMgr_CompactFreeSpace=n**
  - Default: 50%, Compacts database when specified percentage is free
- **CertMgr_CompactDays=n**
  - Default: 30, Compacts database when not compacted since number of days
- **CertMgr_ACCEPT_TOU=1**
  - Same as command-line option to confirm ACME provider terms of use – useful for automation

Common Issues & Tracing

- ACME license terms not accepted
- Port 80 cannot be reached – DNS or Firewall issue
- Most errors are already visible in TLS Credentials document
  - More detailed information can be found in debug logs if enabled
- DNS-TXT provider cannot be reached or configuration problem
  - DNS provider trace should be set to **error logging** by default



Debugging and Troubleshooting Command Line

-v = Verbose logging (log.nsf)

-d = Debug mode

- IBM_TECHNICAL_SUPPORT/certmgr_debug_[..].log

-l = Log all Curl I/O to file

- IBM_TECHNICAL_SUPPORT/certmgr_curl__[..].log

-z = Connectivity test: Just get the ACME directory URLs and terminate

- Useful for testing internet connectivity

- Example: **load certmgr -d -l**

TLS Cache Logging and Debugging

- **CERTSTORE_CACHELOG=1**
  - Recommended Starting point for all troubleshooting
  - Logs most important events only

- **CERTSTORE_CACHELOG=2**
  - Very detailed logging → Debug mode

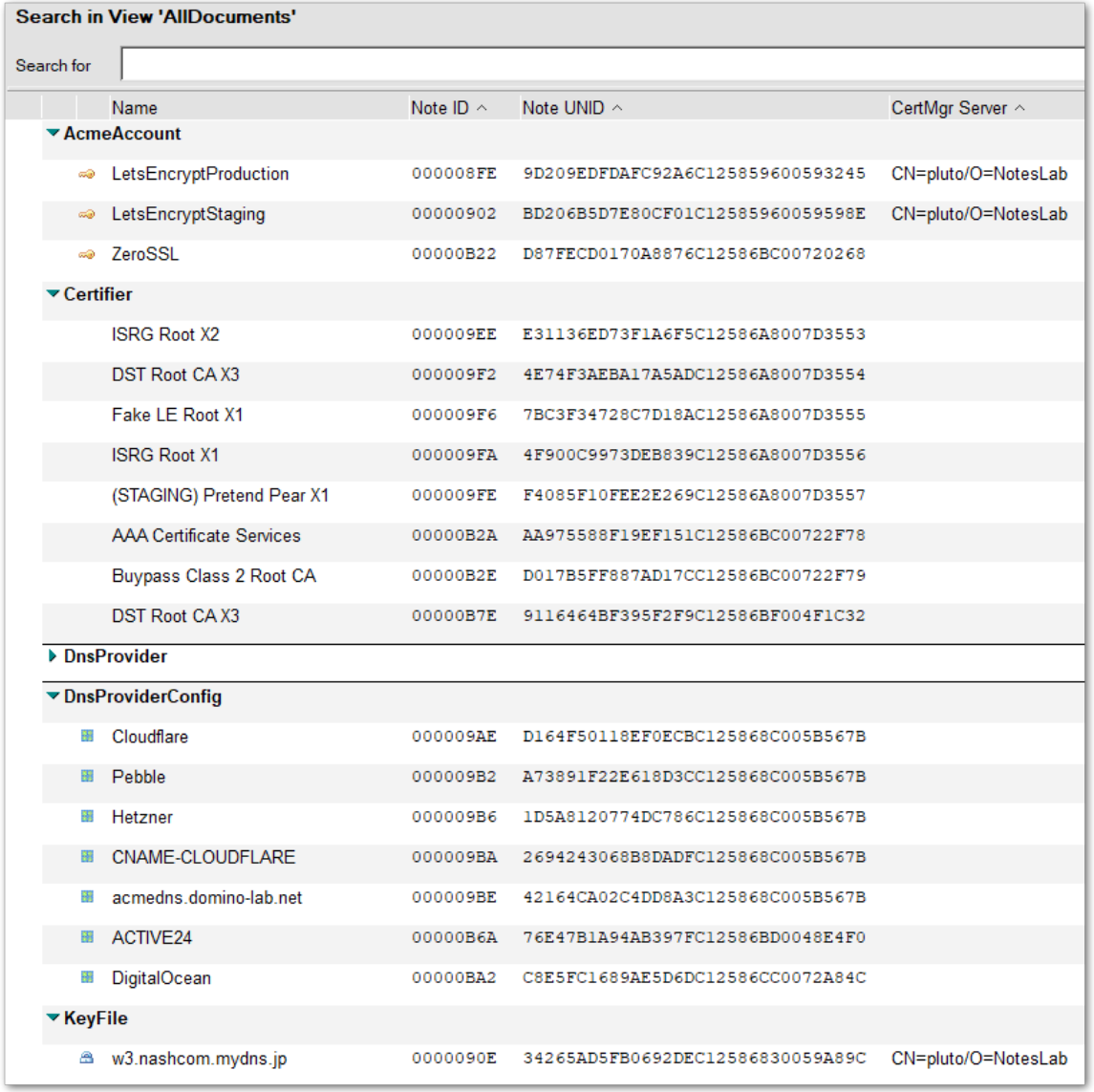
- **DEBUG_SSL_TLSCACHE=1 *)**
  - Debug SSL side of TLS Cache
- **DEBUG_SSL_KYRCACHE=2 *)**
- Debug SSL for old KYR Cache

*) Task restart needed

Hidden AllDocuments View

- Open via CTRL+Shift → View → Goto
- This view contains all documents by form
- CertMgr Server is listed for all documents encrypted

- Secondary sort column by NoteID and NoteUNID
  - Find documents listed in low level error messages



HCLSoftware