[Certificate Creation 2](#_Toc478840822)

[CA Certificate 2](#_Toc478840823)

[Server Certificate 2](#_Toc478840824)

[Key Usages 2](#_Toc478840825)

[CommonName 2](#_Toc478840826)

[Client Certificate 2](#_Toc478840827)

[Key Usages 2](#_Toc478840828)

[CommonName 2](#_Toc478840829)

[Useful Environment Variables 2](#_Toc478840830)

# Certificate Creation

## CA Certificate

Should be exported with entire CA chain to work properly.

## Server Certificate

### Key Usages

NSCertType: SSL Server

Key Usage: Digital Signature, Key Encipherment

Extended Key Usage: TLS Web Server Authentication, IP Sec End System

### CommonName

openvpn.jgelinas.com OpenVPN Server

## Client Certificate

### Key Usages

NSCertType: SSL Client

Key Usage: Digital Signature, Key Agreement

Extended Key Usage: TLS Web Client Authentication, IPSecUser

### CommonName

OpenVPN *name of client* Client

ie – OpenVPN Jean Gelinas Client

# Useful Environment Variables

## tls\_digest\_# -

usually tls\_digest\_0 for the certificate related to the client

gives the SHA1 fingerprint of the client cert

set prior to execution of --tls\_verify script

## tls\_digest\_sha256\_#

usually tls\_digest\_sha256\_0 for the certificate related to the client

gives the SHA256 fingerprint of the client cert

set prior to execution of --tls\_verify\_script

## common\_name

X509 common name of an authenticated client

set prior to execution of --client-connect, --client-disconnect and –auth-user-pass-verify scripts

## signal

reason for exit or restart of the client (sigurs1, sighup, sigterm, sigint, inactive, ping-exit, ping-restart, connection-reset, error, unknown)

set prior to --down script execution.

## X509\_n\_CN

common name of certificate used to connect with , n=0 usually the actual client certificate

In my installation n = 0 would be OpenVPN Jean Gelinas Client.

In my installation n = 1 would be openvpn.jgelinas.com OpenVPN CA

## time\_ascii

client connection timestamp

set prior to execution of --client\_connect script.

## trusted\_ip

actual ip address of connection client

set prior to execution of --ipchange, --client-connect, --client-disconnect scripts

## trusted\_port

actual port used for connection client

set prior to execution of --ipchange, --client\_connect, --client\_disconnect scripts

## username

username provided by a connecting client

set prior to --auth-user-pass-verify when the via-env modifier is specified

don’t worry, can be use in addition to --auth-verify

could be used to check a username from a database as well as using the certs to verify

# Commands That Trigger Scripts

## --auth-user-pass-verify *cmd*

cmd should be via-env where the password and the username are passed via environment variables.

can be used in addition to --auth-verify, to check a database for username/password combos from a database, perhaps.

# Setting Up Dual Factor Authentication

* choose a smartcard provider where you can load your private client key and client certificate. (feitian)
* the smartcard/token provider will take care of the ‘what you know’ aspect of the security by requiring you enter a password/pin to access the key and certificate within the smartcard
* the smartcard provider will provide the means to load your private key and certificate
* set up OpenVPN with the driver for the smartcard
  + This would be the file location that you care about for a Feitian EPS2003 smart card usb C:/Windows/System32/eps2003csp11.dll
* once this driver is loaded, OpenVPN will search the smartcard (I’m assuming that the smartcard needs to be loaded onto your OpenVPN machine at this point) and gather the serialized id that represent your private key and certificate. Believe the provider should provide a ‘pam’ module to gather the necessary PIN to unlock the key and certificate
* set up the driver provider location and the serialized id from your smartcard in your client config
  + it is probably best to ‘push’ this information to your client in the --tls-verify script and the --auth-user-pass-verify script
    - --push -auth-token

## Viewing the contents of the smartcard to get the serial number

$ openvpn --show-pkcs11-ids C:/Windows/System32/eps2003csp11.dll

will show:

Certificate

DN:

CN=movpn

Serial: 01

Serialized id: SafeNet\x20Inc\x2E/eToken/00a3659e/Mastering\ x20OpenVPN/20141001

00s3659e – the serial number of the token

Mastering OpenVPN – the name of the token

20141001 – id of the private key and the certificate

## What goes into the config file

pkcs11-providers C:/Windows/System32/eps2003csp11.dll

pkcs11-id 'SafeNet\x20Inc\x2E/eToken/00a3659e/Mastering\ x20OpenVPN/20141001’

# Verifications Done In Config File Based On Certificate Field Values

## Require that client certificate have certain key usages

**--remote-cert-tls client –** means that the cert must have TLS Web Client Authentication.

can also use:

**--remote-cert-ku 544c532057656220436c69656e742041757468656e7469636174696f6e**

only difference here is that “TLS Web Client Authentication” must be in hex, not plain string.

can also use:

**--remote-cert-eku oid**

ie –remote-cert-eku 1.3.6.1.5.5.7.3.2 (TLS Web Client Authentication)

can also use:

**--verify-x509-name name “subject”**

requires that the entire subject string match

ie - --verify-x509-name ‘emailAddress=jgelinas3333@gmail.com,CN=openvpn.jgelinas.com OpenVPN Server,OU=openvpn.jgelinas.com,O=jgelinas.com,L=Manchester,ST=NH,C=US’

# OpenSC

DF – directory files – directory structure of PKCS#15

EF – elementary files – the actual files of PKCS#15