Tempo2Market Manufacturing Requirements for Identification Branding at the End-Of-Line (EOL)

1 General info

This documentation describes requirements for the Manufacturing-End-Of-Line (EOL) setup.

So that an unspecific embedded device (unit) gets its specific "branding" (signed key).

This branding gets utilised in the field for general authentication (like cloud services) and copy-protection.

For more info regarding the REST-API which is served by the unit its branding_mini_server.elf, see yaml documentation "T2M Identification branding server REST-Api" (filename: brandingserver.yaml).

2 What is provided?

To provide you with an **intermediate CA certificate**, you need to send us a CSR. This CSR will be sigend by our T2M-ROOT-CA so that the unit can use our (cloud) services.

If you want to deploy your own signature, use your own root CA here.

3 EOL Software Entity Requirements

1. DHCP network with mini dns setup so that the unit gets its network connection
2. An EOL setup software (e.g. a script) that talks to the Rest-API, see next chapter

4 EOL setup software scheme: sequence of operation

This is what you need to implement:

1. Flash T2M Bootloader + Initramfs into unit and reboot, unit will then detect its initial status, will get an IP from the DHCP and serves a REST-API
2. Poll for availability of REST-API, usage of DHCP info could be necessary to get correct IP
3. Call: "get newcsr" to get a csr from the unit, here you set e.g. your internal per device product id as CN (see yaml docu for more info).
4. Sign the csr with your intermediate CA certificate
5. Call: "put certificate"

Mind the respone codes.

Please use elliptic curve cryptography (ECC) based on NIST-P256 curve with at least SHA256 or a more reliable hash algorithm.

5 Identification Branding Procedure

5.1 Assumptions:

- A new device is produced by `"ExampleOEM"` in Germany and it is assigned the unique serial number `"Example123456"`.
- The device has no private key and no certificate yet.
- The newly manufactured device is connected to a network and has been assigned the arbitrary IPv4 address 10.0.1.20 with subnet mask 255.255.0.0.
- The `ExampleOEM's` CSR for a T2M Production Intermediate CA has been signed by the T2M Root CA and the production intermediate certificate is named `OEM_T2M_PROD_CA_01`.
- The PC holding the CA certificate is well secured by an audited process and it is connected to the same network as the device.

5.2 Steps to follow

5.2.1 Install/Flash/Boot the T2M production image software

The `ttmdaemon` application will run with the argument `-branding`. The ttmdaemon process will create an HTTP server to service the REST-API for identification branding documented in **brandingserver.yaml**.

5.2.2 Send a GET request to `http://10.0.1.20/rest/newcsr` to get a CSR of the device

This request will ask the device to create a new ECC private key and issue a certificate signing request (CSR). In case that the device has no certificate installed, the server will respond with status `200 OK` and return the CSR of the device as `"text/plain"` content. The CSR produced by the device will contain a subject distinguished name like this:

`"C=DE,O=ExampleOEM,CN=Example123456,DC=367820ef9c9d2ef9c9d299c5980d5..."`

Example:

```
curl -X GET "http://10.0.1.20/rest/newcsr" \
    -o device.csr \
    -H  "Content-Type: application/json" \
    -d "{  \"CN\": \"Example123456\",  \"O\": \"ExampleOEM\",  \"C\": \"DE\"}"
```

5.2.3 Sign the CSR with the production CA certificate `OEM_T2M_PROD_CA_01`

We recommend the use of OpenSSL with a configuration file in order to sign a device certificate with the production CA. See below an example of an openssl configuration file for the production CA. In the following example configuration we assume that the private key of the production CA `OEM_T2M_PROD_CA_01` is stored in `/home/admin/private.key` and the CA's X.509 certificate is stored in `/home/admin/t2mprodca01.crt`.

Example OpenSSL intermediate CA configuration file intermediate.cnf

```
[ ca ]
# `man ca`
default_ca = CA_default

[ CA_default ]
# Directory and file locations.
dir               = /home/admin
certs             = $dir/certs
crl_dir           = $dir/crl
new_certs_dir     = $dir/newcerts
database          = $dir/index.txt
serial            = $dir/serial
RANDFILE          = $dir/private/.rand

# The CA key and certificate.
private_key       = $dir/private.key
certificate       = $dir/t2mprodca01.crt

# For certificate revocation lists.
crlnumber         = $dir/crlnumber
crl               = $dir/crl/intermediate.crl.pem
crl_extensions    = crl_ext
default_crl_days  = 30

default_md        = sha256

name_opt          = ca_default
cert_opt          = ca_default
```

```
default_days      = 2920
preserve          = no
policy            = policy_loose

[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName             = supplied
stateOrProvinceName     = optional
localityName            = optional
organizationName        = supplied
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = client, email, objsign
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always
```

To sign the CSR a validity period of the certificate must be provided. We recommend to issue the device certificates for four to eight years.

The following openssl command can be used to sign a device CSR stored in /home/admin/device.csr for 6 years:

```
openssl ca -config /home/admin/intermediate.cnf -extensions usr_cert \
    -days 2190 -notext -md sha256 \
    -in /home/admin/device.csr \
    -out /home/admin/device.x509.cert
```

5.2.4 Create the device certificate chain

Concatenate the signed device certificate and the production CA certificate to build a certificate chain that can be verified by the T2M cloud.

The following shell command can be used to concatenate the certificates

```
cat /home/admin/device.x509.cert \
    /home/admin/t2mprodca01.crt > /home/admin/device.chain.cert
```

The following snippet shows how a chain file of a test device looks like

```
-----BEGIN CERTIFICATE-----
MIICLjCCAdSgAwIBAgICEAIwCgYIKoZIzj0EAwIwRDELMAkGA1UEBhMCREUxCzAJ
BgNVBAgMAkJXMQ8wDQYDVQQKDAZIZXhkZXYxFzAVBgNVBAMMDlQyTV9BVVRIX0NB
XzAxMB4XDTE5MDkzMDEzNTQyNloXDTIwMDEyODEzNTQyNlowNDELMAkGA1UEBhMC
REUxDTALBgNVBAoMBFRlc3QxFjAUBgNVBAMMDVNlcmlhbER1bW15MDEwWTATBgcq
hkjOPQIBBggqhkjOPQMBBwNCAASkIU446YRWkqjZcbl34RFXmAUxIEB7Q+jeP1sr
bywVvBT5dfLfJmtm/SrCEGYQlwupCiBMuPs8/RNxMQBIJSZjo4HFMIHCMAkGA1Ud
EwQCMAAwEQYJYIZIAYb4QgEBBAQDAgWgMDMGCWCGSAGG+EIBDQQmFiRPcGVuU1NM
IEdlbmVyYXRlZCBDbGllbnQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFOTpf8MqO2+i
```

```
p7CrXzL13nUMwJM+MB8GA1UdIwQYMBaAFNbRrRMRXg487QUzB13zME9u0WI4MA4G
A1UdDwEB/wQEAwIF4DAdBgNVHSUEFjAUBggrBgEFBQcDAgYIKwYBBQUHAwQwCgYI
KoZIzj0EAwIDSAAwRQIhALnoM98A9CBO/Ss/a9AuQNTKG8D+4LmQ7GKiZBtXSoqY
AiBqZJxjyucZCimMRLqzlUD8XP0yQlyjPhKZ/V9MA/x94g==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIB7zCCAZWgAwIBAgICEAAwCgYIKoZIzj0EAwIwVTELMAkGA1UEBhMCREUxCzAJ
BgNVBAgMAkJXMRIwEAYDVQQHDAlTdHV0dGdhcnQxDzANBgNVBAoMBkhleGRldjEU
MBIGA1UEAwwLVDJNX1JPT1RfMDEwHhcNMTkwOTIzMTU0MzQ0WhcNMjkwOTIwMTU0
MzQ0WjBEMQswCQYDVQQGEwJERTELMAkGA1UECAwCQlcxDzANBgNVBAoMBkhleGRl
djEXMBUGA1UEAwwOVDJNX0FVVEhfQ0FfMDEwWTATBgcqhkjOPQIBBggqhkjOPQMB
BwNCAAQwBR0QEUnmGNzX7LqE17tDLbVjhgOuoo3+lQ/zbRD/bqxHbw896F1ipvbl
82ABjtRZMqWeQne1JFhdm1vS2tgwo2YwZDAdBgNVHQ4EFgQU1tGtExFeDjztBTMH
XfMwT27RYjgwHwYDVR0jBBgwFoAUV9Lz6lp5XOoi6EkDChjsdjdLaSIwEgYDVR0T
AQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMCAYYwCgYIKoZIzj0EAwIDSAAwRQIh
AMWMp3welis6n85u+6DCeT0UGRyiC2erzsOI2UYeugYtAiByRmulvya1wFcw0Pi6
ABZ91ajLjngGVNEkTWLi9n13EQ==
-----END CERTIFICATE-----
```

5.2.5 Setup the device with the newly created certificate chain

To setup the device with the newly created certificate chain a JSON request must be sent to

`http://10.0.1.20/rest/certificate`.

The format of the request is described in **brandingserver.yaml**.

The following example constructs a JSON request and sends it to the device

```
(echo -ne '{"chain":"'; cat /home/admin/device.chain.cert; echo -ne '"}') | \
     sed ':a;N;$!ba;s/\n/\\n/g' > \
    /tmp/request.json

curl -X PUT "http://10.0.1.20/rest/certificate" \
    -H  "Content-Type: application/json" \
    -d @/tmp/request.json
```

Note
An optional list of servers may be provided to setup alternative endpoints for server communication.
