PKI Hands-On 1.03

Last updated	18-Jun-2025
Doc. version	1.03
Doc. owner	Uzi Golan
Approved by	
Customer	R&D
Project or installation name	
Project number	
Solution name	Choose an item.
RAD products and versions included	
Content type	
Keywords	

This document contains information that is proprietary to RAD Data Communications Ltd. ("RAD"). No part of this document may be reproduced or published or used in any form whatsoever without prior written approval by RAD Data Communications. Right, title and interest, all information, copyrights, patents, know-how, trade secrets and other intellectual property or other proprietary rights relating to this document and to the products described therein and any software components contained therein are proprietary products of RAD protected under international copyright law and shall be and remain solely with RAD. The trade names mentioned in this document are owned by RAD. No right, license, or interest to such trademark is granted hereunder, and you agree that no such right, license, or interest shall be asserted by you with respect to such trademark. You shall not copy, reverse, compile or reverse assemble all or any portion of this document or the product mentioned therein. You are prohibited from, and shall not, directly or indirectly, develop, market, distribute, license, or sell any product that supports substantially similar functionality to the product mentioned in this document, based on or derived in any way from such a product. Your undertaking in this paragraph shall survive perpetually.

Contents

1 Introduction

1.1 Glossary

SCEP - SCEP (Simple Certificate Enrollment Protocol) enables automated certificate management by allowing devices to securely request and retrieve certificates from a CA using HTTP-based communication.

Quantum-Safe OpenSSL Provider - oqsprovider is an open source OpenSSL provider developed as part of the Open Quantum Safe project.It integrates post quantum cryptographic algorithms into OpenSSL 3.x, enabling applications to generate and use quantum safe keys and operations seamlessly. This provider offers implementations of various

MQTTs Broker - An MQTTs broker is a server that routes messages between clients using a publish/subscribe model and TLS Certificate authentication and encapsulation method.

EST Enrollment method - EST enrollment technology automates certificate issuance by allowing devices to submit CSRs and receive signed certificates over secure channels.

CRL - (Certificate Revocation List) is a mechanism for maintaining and distributing a list of digital certificates that have been revoked by a Certificate Authority, ensuring that clients can verify certificate validity.

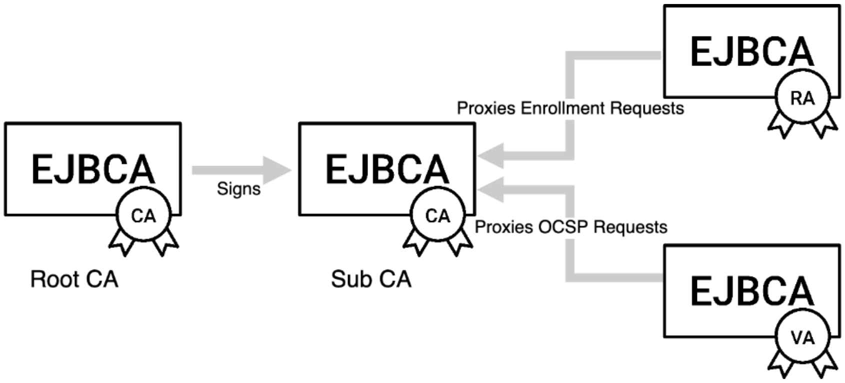
Certificate Revoking - Certificate revoking is the process of invalidating a digital certificate before its expiration, ensuring it can no longer be trusted.

OCSP - A real-time certificate status checking protocol that allows clients to verify whether a digital certificate has been revoked, without downloading the full CRL. OCSP improves performance and bandwidth usage compared to traditional revocation lists.

1.2 PKI Concept

Certification Authority (CA) part of Public Key Infrastructure (PKI) according to standards such as X.509 and IETF-PKIX

PKI Architecture



From : https://docs.keyfactor.com/ejbca/9.0/ejbca-concepts

Root CA

A RootCA has a self-signed certificate and is also called Trusted Root. Verification of other certificates in the PKI ends with the RootCAs self-signed certificate. Since the RootCAs certificate is self-signed it must somehow be configured as a trusted root for all clients in the PKI.

Sub CA

A subordinate CA, or SubCA for short, is a CA whose certificate is signed by another CA, which can be another SubCA or a RootCA. Since the SubCAs certificate is signed by another CA, it does not have to be configured as a trusted root. It is part of a certificate chain that ends in the RootCA.

Registration Authority (RA)

A Registration Authority (RA) is an administrative function that registers entities in the PKI. The RA is trusted to identify and authenticate entities according to the CAs policy. There can be one or more RAs connected to each CA in the PKI.

Validation Authority (VA)

A Validation Authority (VA) is responsible for providing information on whether a certificate is currently valid or not. The VA does not issue or revoke certificates, but it validates certificates by providing a list of revoked certificates for a CA, known as a Certificate Revocation List (CRL). Another method that the VA can support is the Online Certificate Status Protocol (OCSP). It is a real-time lookup of a certificate status, compared to the CRL which is generated on a set schedule. The VA can respond to OCSP requests and reply if a certificate is good, revoked, or unknown. There can be one or more VAs connected to each CA in the PKI.

1.3 Overview

This certificate management system offers an intuitive UI and endpoints to submit CSRs and manage certificates—allowing users to view, download, revoke, and delete them. It also provides access to CA chains, CRLs, and certificate statuses, while supporting advanced protocols like SCEP, EST, and OCSP for comprehensive lifecycle management.

Features

The platform supports the following features either UI or API methods

CA General	Download CA Chain	curl -k https://pikachu-ca.iot-rad.com:5443/downloads/chain	Returns full CA chain file
	Download CRL	curl -k https://pikachu-ca.iot-rad.com:4443/downloads/crl	Generates and downloads latest Certificate Revocation List
	Certificate Status	curl -k https://pikachu-ca.iot-rad.com:4443/status/0xc75f573d9cb2b581	Returns status as valid, revoked, or not found in JSON
	Expired Certificates	curl -k https://pikachu-ca.iot-rad.com:5443/expired	Returns list of certificate IDs that are expired
	Download CSR	curl -k https://pikachu-ca.iot-rad.com:5443///requests/1/download	Serves saved CSR if available
SCEP	SCEP CRL	sscep getcrl -d -u http://pikachu-ca.iot-rad.com:8090/scep -c ca_rsa.crt -w crl.pem -l local.crt -k local.key	
	SCEP Enrolment	sscep enroll -d -v -u http://pikachu-ca.iot-rad.com:8090/scep -c rad_ca_sub_rsa.crt -k client1.key -r client1.csr -l client1.crt	
	SCEP CA Certs	sscep getcap -d -u http://pikachu-ca.iot-rad.com:8090/scep -c cap.pem	

EST	EST Enrollment	curl -k -X POST --data-binary @etx.csr.der https://openxpki.iot-rad.com:4443/.well-known/est/simpleenroll -H "Content-Type: application/pkcs10" --output etx.crt.p7	Accepts DER CSR and returns signed certificate in PKCS#7
	EST CA Certs	curl -k https://pikachu-ca.iot-rad.com:4443/.well-known/est/cacerts --output chain.crt	Returns CA chain in PKCS#7 format
OCSP	OCSP Responder	openssl ocsp -reqout ocsp_request.der -issuer rad_ca_sub.crt -cert valid.crt -url https://pikachu-ca.iot-rad.com:4443/ocsp -resp_text -respout ocsp_response.der	Returns OCSP status for a given certificate in DER format
CA General	Download CA Chain	curl -k https://pikachu-ca.iot-rad.com:5443/downloads/chain	Returns full CA chain file
	Download CRL	curl -k https://pikachu-ca.iot-rad.com:4443/downloads/crl	Generates and downloads latest Certificate Revocation List
	Certificate Status	curl -k https://pikachu-ca.iot-rad.com:4443/status/0xc75f573d9cb2b581	Returns status as valid, revoked, or not found in JSON
	Expired Certificates	curl -k https://pikachu-ca.iot-rad.com:5443/expired	Returns list of certificate IDs that are expired
	Download CSR	curl -k https://pikachu-ca.iot-rad.com:5443///requests/1/download	Serves saved CSR if available

Quantum safe keys

Quantum safe keys are encryption keys generated using algorithms that resist attempts by quantum computers, helping secure data against emerging quantum threats.

For instance,

- **mldsa44** – which corresponds to NIST Level 1 (Dilithium2)
- **mldsa65** – corresponding roughly to Dilithium3 (NIST Level 3)

- **mldsa87** – corresponding roughly to Dilithium5 (NIST Level 5)

1.4 Purpose

The purpose of this platform is to provide a robust PKI solution that securely provisions and manages RAD devices and servers—such as those operating with MQTT—by seamlessly integrating with CA servers.

1.5 Scope

in non-production environments, with a strong emphasis on tenant isolation, robust security measures, and high availability.

2 PKI platform

The Lifecycle maintenance ansible playbook contains files and ansible playbooks. Once installed the user can perform the maintenance operations.

2.1 Server OS requirements

Operating System	Version
Rocky	9.x

2.2 Installation

Python

Install python 3 and complimentary packages

```
sudo dnf install -y epel-release
sudo dnf module enable -y python3.11
sudo dnf install -y python3.11 python3.11-devel python3.11-pip git

sudo dnf groupinstall -y "Development Tools"


sudo alternatives --install /usr/bin/python3 python3 /usr/bin/python3.11 100
sudo alternatives --install /usr/bin/pip3 pip3 /usr/bin/pip3.11 100

pip install flask cryptography oscrypto asn1crypto flask_sqlalchemy
```

allow python39 programs to use privileged ports (below 1024)

```
sudo setcap 'cap_net_bind_service=+ep' /usr/bin/python3.11
```

Server

Extract the pki_server_102.tar.gz

```
tar xvfz pki_server_102.tar.gz
cd pki-server-2
```

it will open the following directories and files

```
├── mqtt
│    ├── clients_certs
│    ├── docker-compose.yml
│    ├── mosquitto
│    └── mosquitto.conf
├── pki-certs
│    ├── ca.cert.pem
│    ├── cert_3.pem
│    ├── cert.csr
│    ├── crl_empty.pem
├── pki-ocsp
│    ├── ca_00.pem
│    ├── ca_02.pem
│    ├── ca1.pem
│    ├── ca.cert.pem
│    ├── ca-intermediate.pem
│    ├── cert_2.pem
│    ├── cert_revoked.pem
│    ├── ocsp.cert.pem
│    ├── ocsp.csr
│    ├── ocsp.key.pem
│    ├── ocsp.py
│    ├── ocsp_request.der
│    └── ocsp_response.der
├── pki-root
│    ├── ca_cert.pem
│    ├── ca_cert.srl
│    ├── ca_key.pem
│    ├── rad_ca_root.crt
│    └── root.cert.pem
├── pki-server-2
│    ├── app.py
│    ├── extensions.py
│    ├── x509_keys.py
│    ├── config.ini
│    ├── x509_requests.py
│    ├── x509_profiles.py
│    ├── asn1.py
│    ├── builders.py
│    ├── config_storage.py
│    ├── dbtypes.py
│    ├── enums.py
│    ├── envelope.py
│    ├── message.py
│    ├── models.py
│    ├── scep.py
│    ├── models.py
│    ├── ca_mode.conf
│    ├── server_ext.cnf
│    ├── certs.db
│    ├── history
│    ├── output.log
│    ├── x509_profiles
│    ├── x509_templates
│    └── html_templates
├── pki-subca
│    ├── rad_ca_sub_ec.key
│    ├── rad_ca_sub_rsa.key
│    ├── rad_ca_sub_ec.crt
│    ├── rad_ca_sub_rsa.crt
│    ├── rad_chain_rsa.crt
│    └── rad_chain_ec.crt
```

Complementary

Following installation of commands and libraries essential for testing and using the server

SSCEP

Rocky 9 lack OS repository installation there for it needed to be complied, linked and installed

```
git clone https://github.com/zhaozg/openscep.git
cd openscep
chmod u+x configure
./configure

make

sudo make install
```

Note : If missing libraries install using the OS dnf

EST-Client

From repo https://github.com/globalsign/est

```
sudo dnf install -y golang

git clone https://github.com/globalsign/est.git

go install github.com/globalsign/est/cmd/estclient@latest

sudo cp go/bin/estclient   /usr/local/bin/estclient
```

Quantum Safe Algorithm

Quantum safe algorithms successful installation requires openssl 3.x and other development packages

Prerequisite installation of Ninja

```
sudo dnf config-manager --set-enabled crb
sudo dnf install -y epel-release
sudo dnf install -y ninja-build

sudo dnf groupinstall -y "Development Tools"
sudo dnf install -y cmake ninja-build git openssl-devel libffi-devel
```

install quantum resistant algorithms (like Dilithium, Falcon, Kyber, and SPHINCS+)

```
git clone https://github.com/open-quantum-safe/oqs-provider.git
cd oqs-provider
./scripts/fullbuild.sh
sudo cmake --install _build
```

find the openssl.cnf file and add it manually

```
ls -l /etc/ssl/openssl.cnf
lrwxrwxrwx. 1 root root 24 Aug 21  2024 /etc/ssl/openssl.cnf ->
/etc/pki/tls/openssl.cnf

# add the new provider

sudo vi /etc/pki/tls/openssl.cnf

[provider_sect]
default = default_sect
oqsprovider = oqsprovider_sect
[default_sect]
activate = 1
[oqsprovider_sect]
activate = 1
```

test the openssl

```
#run command
openssl list -providers
#output
Providers:
  default
    name: OpenSSL Default Provider
    version: 3.2.2
    status: active
  oqsprovider
```

```
    name: OpenSSL OQS Provider
    version: 0.8.1-dev
    status: active
```

we can see that oqsprovider provider is active

MQTTs broker

Using distribution mosquitto with docker compose

```
mkdir mqtt
cd mqtt


cat > docker-compose.yml <<EOL


version: '3'
services:
  mosquitto:
    image: eclipse-mosquitto:latest
    container_name: mosquitto
    network_mode: "host"  # Use host network mode
    ports:
      - "1883:1883"    # Default MQTT
      - "2883:2883"    # Secure MQTT with TLS
      - "9001:9001"    # WebSocket (if needed)
    volumes:
      - ./mosquitto.conf:/mosquitto/config/mosquitto.conf
      - ./mosquitto/certs:/mosquitto/certs
    restart: unless-stopped
EOL
```

make configuration file

```
cat > mosquitto.conf <<EOL

per_listener_settings true

listener 1883 0.0.0.0
allow_anonymous true
log_type all

listener 2883 0.0.0.0
cafile /mosquitto/certs/CA.cert
certfile /mosquitto/certs/est_mqtt_server_1.pem
```

```
keyfile /mosquitto/certs/est_mqtt_server_1.key

crlfile /mosquitto/certs/crl.pem

require_certificate true
use_identity_as_username true
log_type all
EOL
```

populate certification directory

```
mkdir certs
cd certs
```

with files

```
est_mqtt_server_1.pem
est_mqtt_server_1.key
crl_client_2.pem
CA.cert
```

By executing the following command Server is up and running

```
cd pki-srever-2
python app.py
```

all logs are written to the stdout and file output.log

2.3 Generate Root and intermediate certification

Every CA server is based on Root certificate and one or more intermediate (sub) certificates

Following procedure on how to generate a root and intermediate certificates for the CA server

Generate Root Certificate

Create root EC key

```
openssl ecparam -name prime256v1 -genkey -noout -out rad_ca_root.key
```

Prepare certificate configuration file

```
cat > rad_ca_root.cnf <<EOL
# CA Certificate Configuration Template for Root ECC Certificates
[ req ]
default_bits        = 4096
default_md          = sha256
prompt              = no
distinguished_name = dn
x509_extensions     = v3_ca

[ dn ]
C  = IL
ST = TLV
L  = Tel Aviv
O  = RAD
OU = RD
CN = RAD Test ECDSA

[ v3_ca ]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints       = critical, CA:true, pathlen:1
EOL
```

Self-sign root certificate

```
openssl req -config rad_ca_root.cnf -key rad_ca_root.key -new -x509 -days
3650 -sha256 -out rad_ca_root.crt
```

Prepare signing server extension configuration file

```
cat > ca_root_ext.cnf <<EOL
[ v3_intermediate ]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid,issuer
basicConstraints       = critical, CA:true, pathlen:0
keyUsage               = keyCertSign, cRLSign
crlDistributionPoints  = URI:https://pikachu-ca.rnd-rad.com/downloads/crl
EOL
```

Generating Intermediate (Sub) Certificate

EC based Key Certificate

Create sub key

```
openssl ecparam -name prime256v1 -genkey -noout -out rad_ca_sub_ec.key
```

Prepare certificate configuration file

```
cat > rad_ca_sub_ec.cnf<<EOL
[ req ]
default_bits       = 2048
default_md         = sha256
prompt             = no
distinguished_name = dn
req_extensions     = v3_intermediate
[ dn ]
C  = IL
ST = TLV
L  = Tel Aviv
O  = RAD
OU = RD
CN = RADSubTestECDSA
[ v3_intermediate ]
subjectKeyIdentifier   = hash
#authorityKeyIdentifier = keyid,issuer
basicConstraints       = critical, CA:true, pathlen:0
keyUsage               = keyCertSign, cRLSign
EOL
```

Generate Certificate request

```
openssl req -new -config rad_ca_sub_ec.cnf  -key rad_ca_sub_ec.key -out
rad_ca_sub_ec.csr
```

Sign certificate

```
openssl x509 -req -in rad_ca_sub_ec.csr -CA rad_ca_root.crt -CAkey
rad_ca_root.key -CAcreateserial -out rad_ca_sub_ec.crt -days 3650 -sha256 -
extfile ca_root_ext.cnf -extensions v3_intermediate
```

RSA based Key Certificate

Create sub key

```
openssl genpkey -algorithm RSA -out rad_ca_sub_rsa.key -pkeyopt
rsa_keygen_bits:4096
```

Prepare certificate configuration file

```
cat > rad_ca_sub_ec.cnf<<EOL
[ req ]
default_bits       = 4096
default_md         = sha256
prompt             = no
distinguished_name = dn
req_extensions     = v3_intermediate
[ dn ]
C  = IL
ST = TLV
L  = Tel Aviv
O  = RAD
OU = RD
CN = RADSubTestECDSA
[ v3_intermediate ]
subjectKeyIdentifier   = hash
#authorityKeyIdentifier = keyid,issuer
basicConstraints       = critical, CA:true, pathlen:0
keyUsage               = keyCertSign, cRLSign
EOL
```

Generate certificate request

```
openssl req -new -config rad_ca_sub_rsa.cnf  -key rad_ca_sub_rsa.key -out
rad_ca_sub_rsa.csr
```

Sign certificate

```
penssl x509 -req -in rad_ca_sub_rsa.csr -CA rad_ca_root.crt -CAkey
rad_ca_root.key -CAcreateserial -out rad_ca_sub_rsa.crt -days 3650 -sha256 -
extfile ca_root_ext.cnf -extensions v3_intermediate
```

Copy keys Certificates

The Certificates and Keys must be located according to the attributes in the CA server configuration file (config.ini)

```
cp  rad_ca_root.key  rad_ca_root.crt  ~/pki-root/
cp rad_ca_sub_ec.crt
rad_ca_sub_rsa.crt  rad_ca_sub_ec.key  rad_ca_sub_rsa.key  ~/pki-subca
cat ~/pki-subca/rad_ca_sub_ec.crt ~/pki-root/rad_ca_root.crt > ~/pki-
subca/rad_chain_ec.crt
cat ~/pki-subca/rad_ca_sub_rsa.crt ~/pki-root/rad_ca_root.crt > ~/pki-
subca/rad_chain_rsa.crt
```

2.4 Configuration

The server is configured by config.ini file

```
[DEFAULT]
# general Flask settings
SECRET_KEY = your-super-secret

[CA]
# Which subordinate CA to use by default: "EC" or "RSA"
mode = EC
# Paths for both modes; the get_ca_config() helper below will pick the right
SUBCA_KEY_PATH_EC    = /home/rocky/pki-subca/rad_ca_sub_ec.key
SUBCA_CERT_PATH_EC   = /home/rocky/pki-subca/rad_ca_sub_ec.crt
CHAIN_FILE_PATH_EC   = /home/rocky/pki-subca/rad_chain_ec.crt
SUBCA_KEY_PATH_RSA   = /home/rocky/pki-subca/rad_ca_sub_rsa.key
SUBCA_CERT_PATH_RSA  = /home/rocky/pki-subca/rad_ca_sub_rsa.crt
CHAIN_FILE_PATH_RSA  = /home/rocky/pki-subca/rad_chain_rsa.crt
ROOT_CERT_PATH       = /home/rocky/pki-root/rad_ca_root.crt

[SCEP]
# enable or disable SCEP entirely
enabled = true

# optional path to a file where we persist the serial across restarts
serial_file = /home/rocky/pki-subca/serial.txt

# Dump directory for raw SCEP requests (optional)
dump_dir = /home/rocky/pki-subca/dumps

# HTTP port for unauthenticated SCEP
http_port = 8090

[HTTPS]
# HTTPS certificate & key for your main CA UI
ssl_cert = /home/rocky/pki-https/tls.cert.pem
ssl_key  = /home/rocky/pki-https/tls.key.pem
port     = 443

[TRUSTED_HTTPS]
# HTTPS certificate & key for your main CA UI
trusted_ssl_cert = /home/rocky/pki-https/pikachu_issued_https.crt
trusted_ssl_key  = /home/rocky/pki-https/pikachu_issued_https.key
trusted_port     = 4443

[PATHS]
# everything else that was hard-coded
crl_path       = crl.pem
server_ext_cfg = server_ext.cnf
validity_conf  = validity.conf
db_path        = certs.db
```

2.5 Run Server

By executing the following command Server is up and running

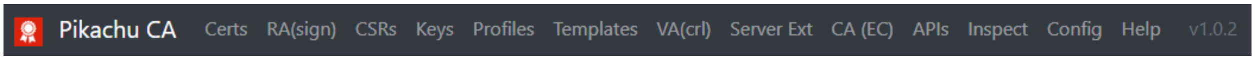
```
cd pki-srever-2
nohup python app.py > app.log 2>&1 &
```

all logs are written to the stdout and file output.log

server can be access using the URL https://pikachu-ca.iot-rad.com:4443

2.6 Layout

Navigate Tiles



Certs - list certificates (Main view)

RA (sign) – Sign certificate

CSR Requests – create, delete, view and list certificates requests

Keys - create, delete, download view and list of keys

Profiles - create, edit, delete, view and list of profiles

Templates - Create Profile based on Templates

VA (crl) – list revoked certificates and updated CRL

Server Ext – view edit and load profile to Extension configuration

CA(mode) – show the root and sub/intermediate certificates details

APIs – list of all APIs download CRL, CA certificate. Enrolments techniques Manual, SCEP and EST, and RA OCSP.

Inspect – Inspect PEM block or Base64-encoded DER data

Config – show content of server config.ini file

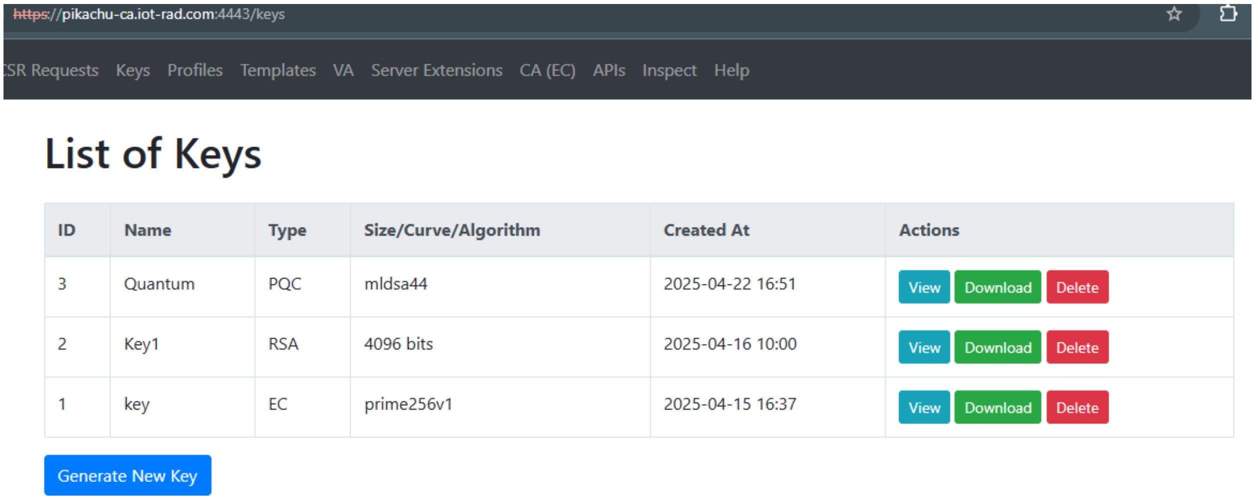
Help – open new browser tab with help Pdf file

Version – 1.0.2

2.7 Generate Key

At the main view under management Links select Keys

Redirected to view with list of Keys


https://pikachu-ca.iot-rad.com:4443/keys
CSR Requests Keys Profiles Templates VA Server Extensions CA (EC) APIs Inspect Help

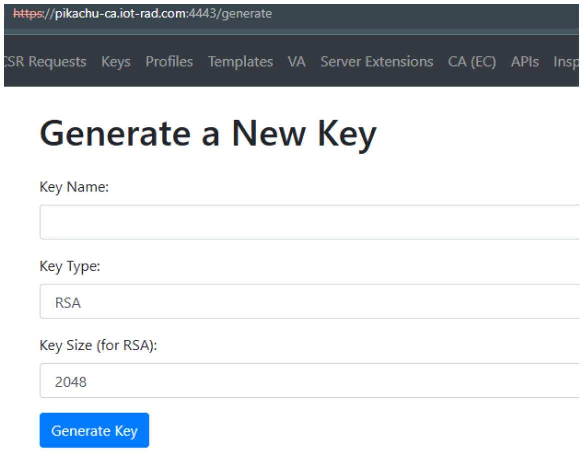
List of Keys

ID	Name	Type	Size/Curve/Algorithm	Created At	Actions
3	Quantum	PQC	mldsa44	2025-04-22 16:51	View Download Delete
2	Key1	RSA	4096 bits	2025-04-16 10:00	View Download Delete
1	key	EC	prime256v1	2025-04-15 16:37	View Download Delete

https://openxpki.iot-rad.com:4443/keys

to generate new key press button Generate New Key

Redirected to Generate a New Key


https://pikachu-ca.iot-rad.com:4443/generate
CSR Requests Keys Profiles Templates VA Server Extensions CA (EC) APIs Insp

Generate a New Key

Key Name:

Key Type: RSA

Key Size (for RSA): 2048

Users can use either RSA, EC (elastic curve) and PQC (post-quantum) Key Type

RSA contains key sizes 2048,3072 and 4096

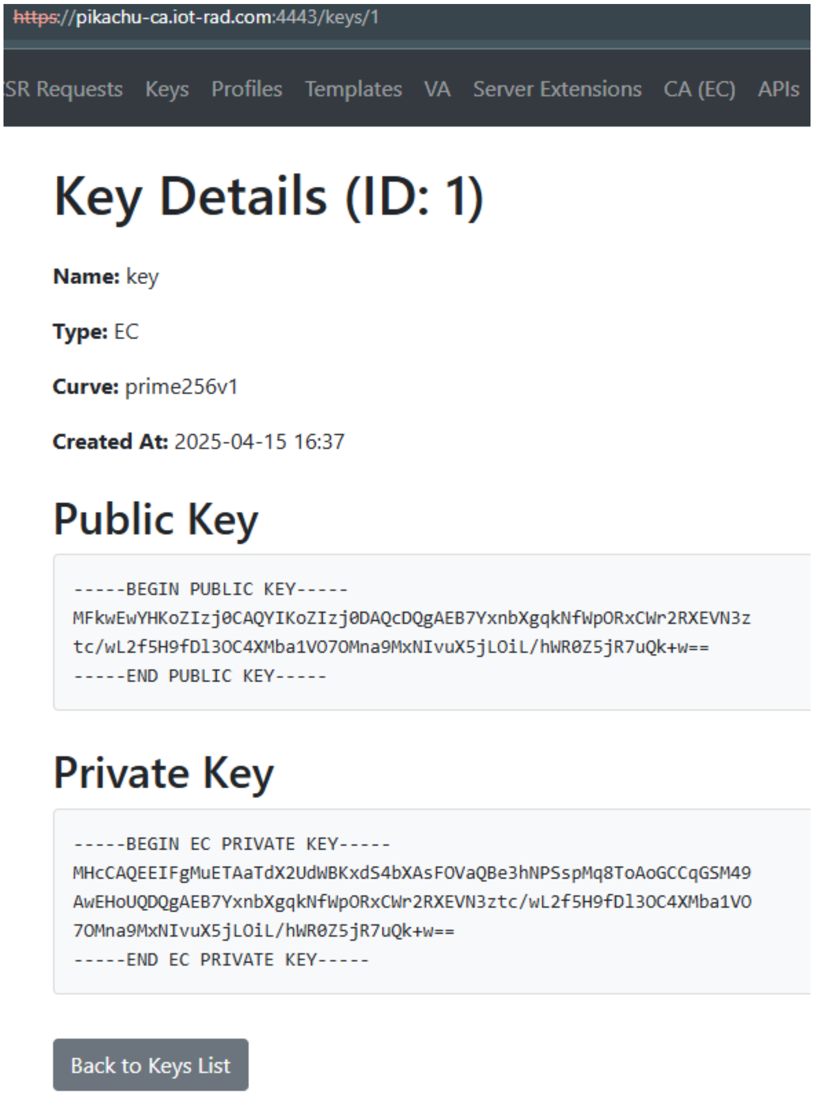
EC has curves prime256v1(secp256r1), secp384r1, secp521r1 and secp256k1

PQC has algorithms mldsa44 (Dilithium2 / NIST L1), mldsa65 (Dilithium3 / NIST L3) and mlds87 (Dilithium5 / NIST L5)

Add a key name and press button generate Key

The view will be redirected to List of Keys view

Press View at the key you wish to see details



Key Details (ID: 1)

Name: key

Type: EC

Curve: prime256v1

Created At: 2025-04-15 16:37

Public Key

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEB7YxnbXgqkNfWpORxCWr2RXEVN3z
tc/wL2f5H9fDl3OC4XMba1VO7OMna9MxNIvuX5jLOiL/hWR0Z5jR7uQk+w==
-----END PUBLIC KEY-----
```

Private Key

```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIFgMuETAaTdX2UdWBKxdS4bXAsFOVaQBe3hNPSspMq8ToAoGCCqGSM49
AwEHoUQDQgAEB7YxnbXgqkNfWpORxCWr2RXEVN3ztc/wL2f5H9fDl3OC4XMba1VO
7OMna9MxNIvuX5jLOiL/hWR0Z5jR7uQk+w==
-----END EC PRIVATE KEY-----
```

User can copy the Private and Public key data for any purpose

2.8 Templates and Profiles Management

The profile is used for various purposes

1. Certificate configuration to generate CSRs
2. CA server extension configuration to sing a CSR

The profile is rendered from template

Templates

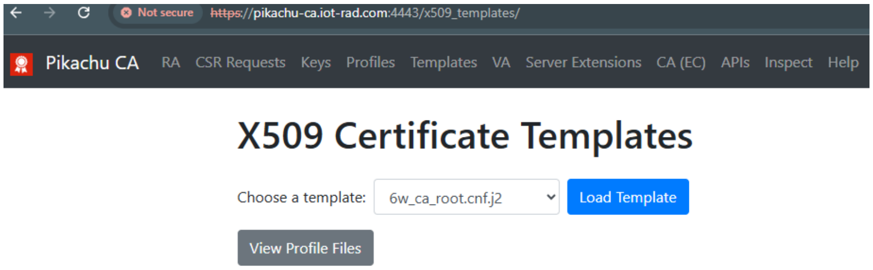
Templates added manually to the server under folder x509_templates

Template is jinja j2 style template for example

```
# CA Certificate Configuration Template for Root ECC Certificates
[ req ]
# Note: For ECC keys the “default_bits” option is not used.
default_md          = sha256
default_days        = 3650
prompt              = no
distinguished_name = dn
x509_extensions     = v3_ca
default_ec_curve    = prime256v1

[ dn ]
C  = {{ ca_country | default("FR") }}
ST = {{ ca_state | default("Ile-de-France") }}
L  = {{ ca_city | default("Paris") }}
O  = {{ ca_organization | default("6WIND") }}
OU = {{ ca_organizational_unit | default("CA Division") }}
CN = {{ ca_common_name | default("6WIND Test ECDSA RCA") }}

[ v3_ca ]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints       = critical, CA:true, pathlen:0
keyUsage               = critical, digitalSignature, keyCertSign, cRLSign
```

At the main view under management Links select Templates

Redirected to view Select an X509 Template



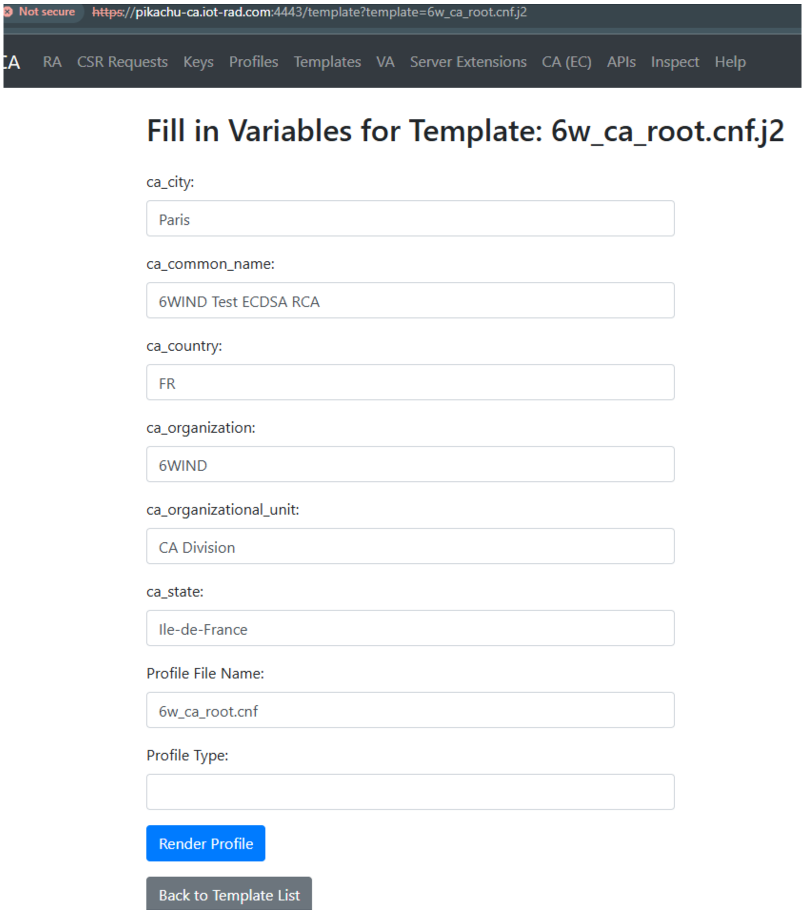
https://openxpki.iot-rad.com:4443/x509_templates/

Choose template and Load template by pressing Load Template button

The view will be redirected to Fill in Variables for Template

Profiles

Change the defaults value to any you wish



Fill in Variables for Template: 6w_ca_root.cnf.j2

ca_city: Paris

ca_common_name: 6WIND Test ECDSA RCA

ca_country: FR

ca_organization: 6WIND

ca_organizational_unit: CA Division

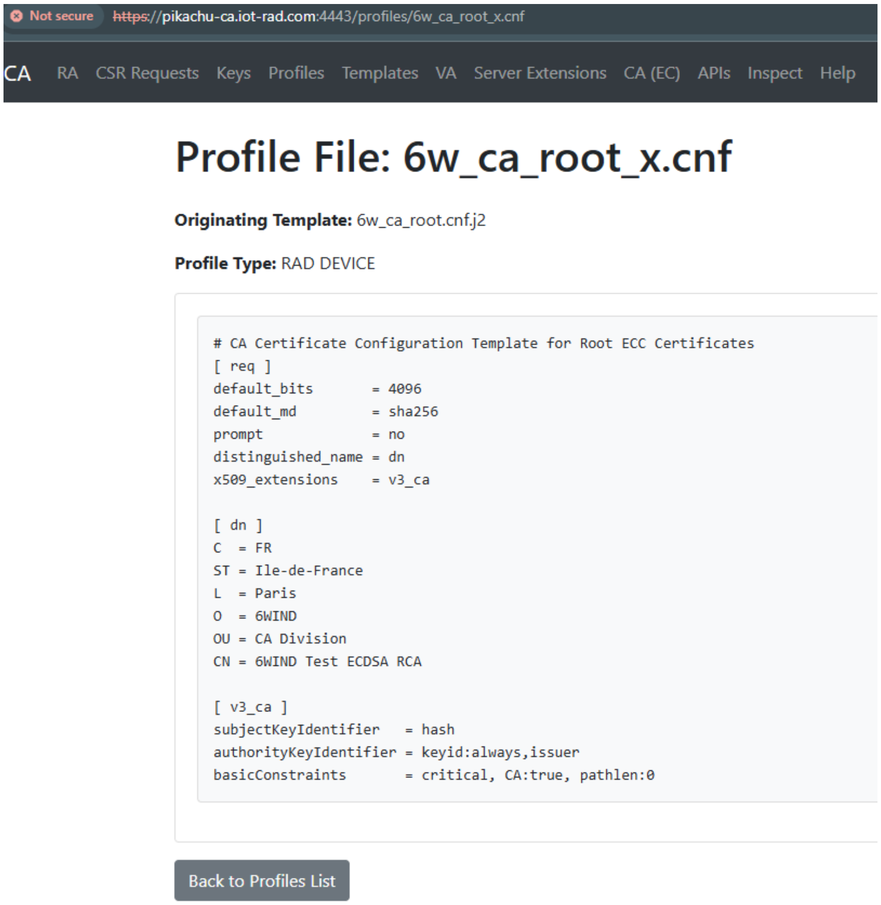
ca_state: Ile-de-France

Profile File Name: 6w_ca_root.cnf

Profile Type:

Press Render profile to create a profile

The view will be redirected to Profile File with generate profile details



Profile File: 6w_ca_root_x.cnf

Originating Template: 6w_ca_root.cnf.j2

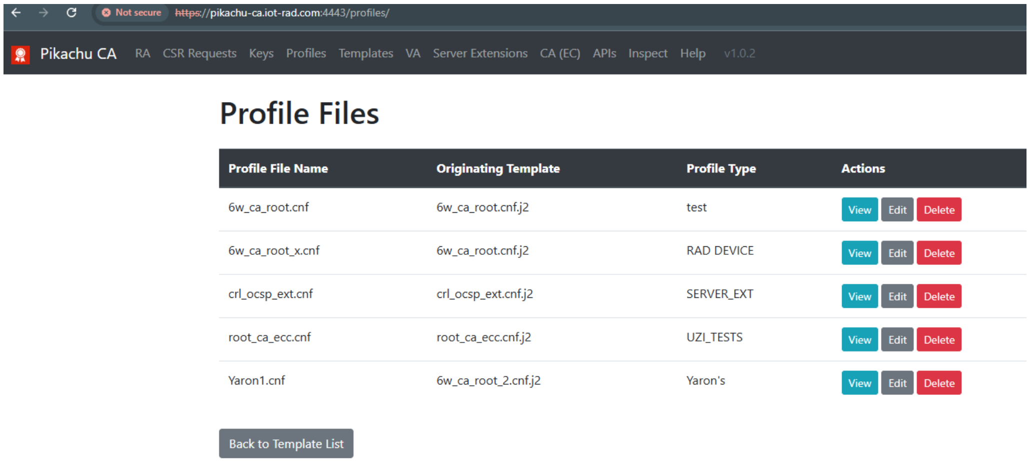
Profile Type: RAD DEVICE

```
# CA Certificate Configuration Template for Root ECC Certificates
[ req ]
default_bits       = 4096
default_md         = sha256
prompt             = no
distinguished_name = dn
x509_extensions    = v3_ca

[ dn ]
C  = FR
ST = Ile-de-France
L  = Paris
O  = 6WIND
OU = CA Division
CN = 6WIND Test ECDSA RCA

[ v3_ca ]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints       = critical, CA:true, pathlen:0
```

A file with the name that has been given created at folder x509_profiles

Users can either press either the “Back to Template List” button or “View All Profiles Files” one

View List of Profile Files contains all profiles able to view, edit or delete

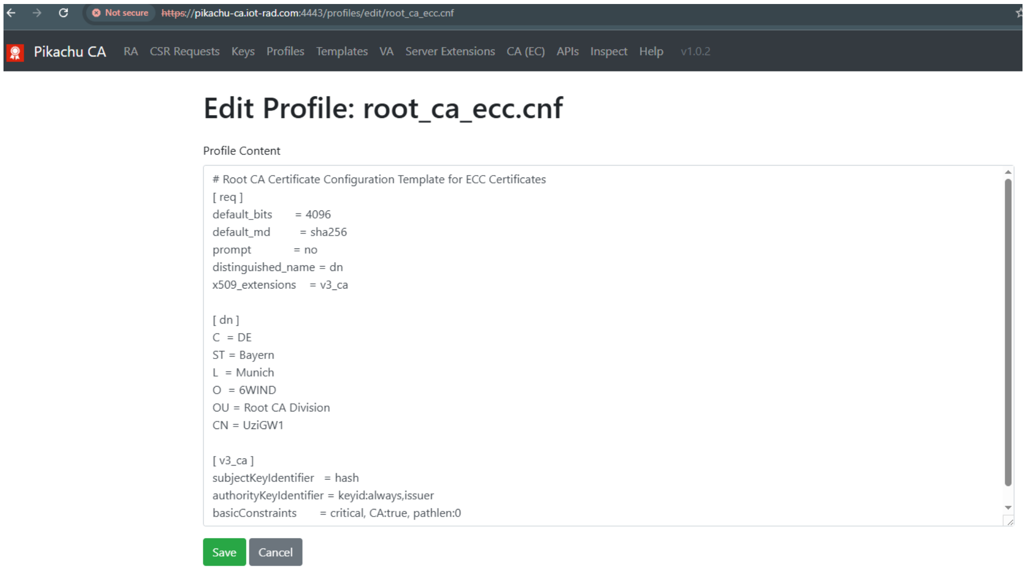


Profile Files

Profile File Name	Originating Template	Profile Type	Actions
6w_ca_root.cnf	6w_ca_root.cnf.j2	test	View Edit Delete
6w_ca_root_x.cnf	6w_ca_root.cnf.j2	RAD DEVICE	View Edit Delete
crl_ocsp_ext.cnf	crl_ocsp_ext.cnf.j2	SERVER_EXT	View Edit Delete
root_ca_ecc.cnf	root_ca_ecc.cnf.j2	UZI_TESTS	View Edit Delete
Yaron1.cnf	6w_ca_root_2.cnf.j2	Yaron's	View Edit Delete

https://openxpki.iot-rad.com:4443/profiles/

to edit profile, use the Edit Action



Edit Profile: root_ca_ecc.cnf

Profile Content

```
# Root CA Certificate Configuration Template for ECC Certificates
[ req ]
default_bits      = 4096
default_md        = sha256
prompt            = no
distinguished_name = dn
x509_extensions    = v3_ca

[ dn ]
C  = DE
ST = Bayern
L  = Munich
O  = 6WIND
OU = Root CA Division
CN = UziGW1

[ v3_ca ]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints       = critical, CA:true, pathlen:0
```

2.9 Certificate actions

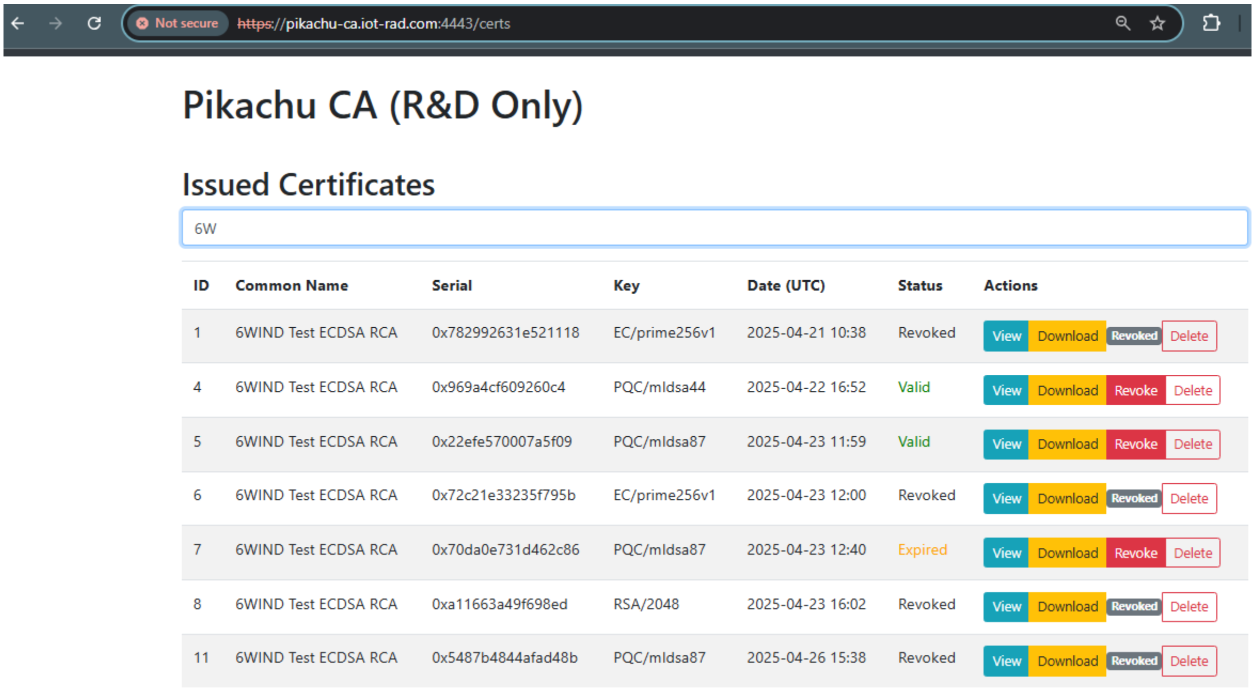
Once certificate has been signed and issued the user can perform the following actions

1. View
2. Download
3. Revoke
4. Delete

Note: Users can also view the Root and intermediate (Sub) Certificate

List

All certificates are contained in the Issued Certificates Table



Pikachu CA (R&D Only)

Issued Certificates

6W

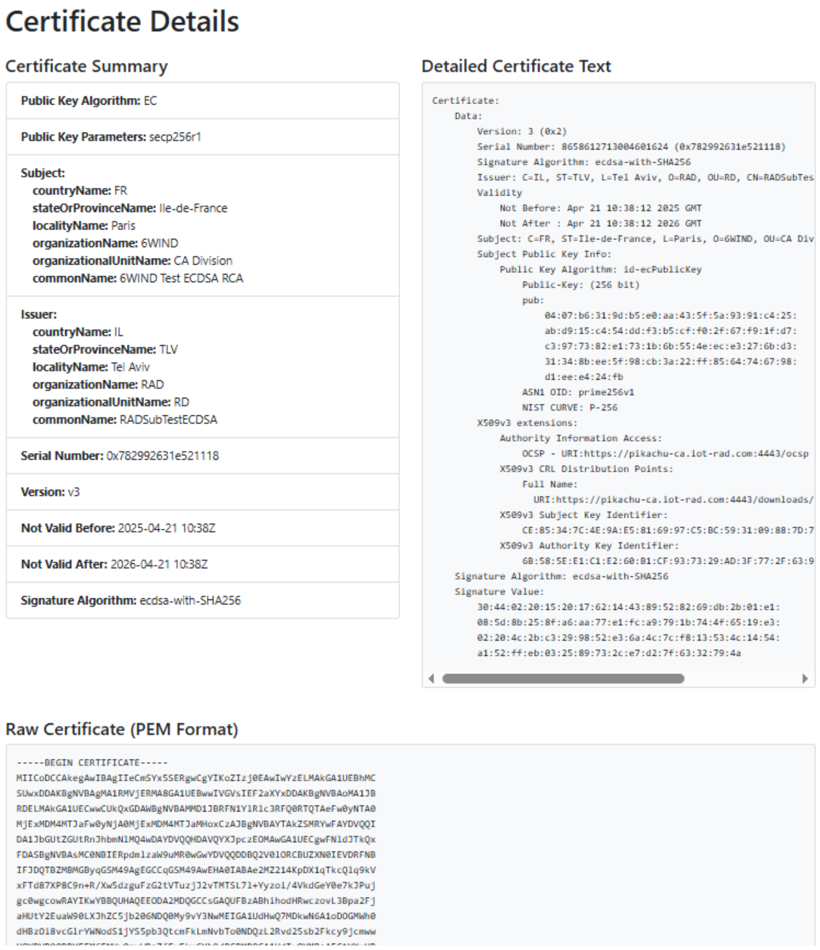
ID	Common Name	Serial	Key	Date (UTC)	Status	Actions
1	6WIND Test ECDSA RCA	0x782992631e521118	EC/prime256v1	2025-04-21 10:38	Revoked	View Download Revoked Delete
4	6WIND Test ECDSA RCA	0x969a4cf609260c4	PQC/mldsa44	2025-04-22 16:52	Valid	View Download Revoke Delete
5	6WIND Test ECDSA RCA	0x22efe570007a5f09	PQC/mldsa87	2025-04-23 11:59	Valid	View Download Revoke Delete
6	6WIND Test ECDSA RCA	0x72c21e33235f795b	EC/prime256v1	2025-04-23 12:00	Revoked	View Download Revoked Delete
7	6WIND Test ECDSA RCA	0x70da0e731d462c86	PQC/mldsa87	2025-04-23 12:40	Expired	View Download Revoke Delete
8	6WIND Test ECDSA RCA	0xa11663a49f698ed	RSA/2048	2025-04-23 16:02	Revoked	View Download Revoked Delete
11	6WIND Test ECDSA RCA	0x5487b4844afad48b	PQC/mldsa87	2025-04-26 15:38	Revoked	View Download Revoked Delete

The table can be filtered by either part of Common Name, Serial values, Key and Date

View

Pressing the View Action open new page (pressing back to home return to the main view)

Change



Certificate Details

Certificate Summary

Public Key Algorithm: EC

Public Key Parameters: secp256r1

Subject:
countryName: FR
stateOrProvinceName: Ile-de-France
localityName: Paris
organizationName: 6WIND
organizationalUnitName: CA Division
commonName: 6WIND Test ECDSA RCA

Issuer:
countryName: IL
stateOrProvinceName: TLV
localityName: Tel Aviv
organizationName: RAD
organizationalUnitName: RD
commonName: RADSubTestECDSA

Serial Number: 0x782992631e521118

Version: v3

Not Valid Before: 2025-04-21 10:38Z

Not Valid After: 2026-04-21 10:38Z

Signature Algorithm: ecdsa-with-SHA256

Detailed Certificate Text

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 8658612713004601624 (0x782992631e521118)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C=IL, ST=TLV, L=Tel Aviv, O=RAD, OU=RD, CN=RADSubTes
        Validity
            Not Before: Apr 21 10:38:12 2025 GMT
            Not After : Apr 21 10:38:12 2026 GMT
        Subject: C=FR, ST=Ile-de-France, L=Paris, O=6WIND, OU=CA Div
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:07:b6:31:9d:b5:e0:aa:43:5f:5a:93:91:c4:25:
                    ab:d9:15:c4:54:dd:f3:b5:cf:f0:2f:67:f9:1f:d7:
                    c3:97:73:82:e1:73:1b:6b:55:4e:ec:e3:27:6b:d3:
                    31:34:8b:ee:5f:98:cb:3a:22:ff:85:64:74:67:98:
                    d1:ee:e4:24:fb
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            Authority Information Access:
                OCSP - URI:https://pikachu-ca.iot-rad.com:4443/ocsp
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:https://pikachu-ca.iot-rad.com:4443/downloads/
            X509v3 Subject Key Identifier:
                CE:85:34:7C:4E:9A:E5:81:69:97:C5:BC:59:31:09:88:7D:7
            X509v3 Authority Key Identifier:
                6B:58:5E:E1:C1:E2:60:B1:CF:93:73:29:AD:3F:77:2F:63:9
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:44:02:20:15:20:17:62:14:43:89:52:82:69:db:2b:01:e1:
        08:5d:8b:25:8f:a6:aa:77:e1:fc:a9:79:1b:74:4f:65:19:e3:
        02:20:4c:2b:c3:29:98:52:e3:6a:4c:7c:f8:13:53:4c:14:54:
        a1:52:ff:eb:03:25:89:73:2c:e7:d2:7f:63:32:79:4a
```

Raw Certificate (PEM Format)

```
-----BEGIN CERTIFICATE-----
MIICoDCCAkegAwIBAgIIeCmSYx5SERgwCgYIKoZIzj0EAwIwYzELMAkGA1UEBhMC
SUwxDDAKBgNVBAgMA1RMVjERMA8GA1UEBwwIVGVsIEF2aXYxDDAKBgNVBAoMA1JB
RDELMAkGA1UECwwCUkQxGDAWBgNVBAMMD1JBRFN1YlRlc3RFQ0RTQTAeFw0yNTA0
MjExMDM4MTJaFw0yNjA0MjExMDM4MTJaMHoxCzAJBgNVBAYTAkZSMRYwFAYDVQQI
DA1JbGUtZGUtRnJhbmNlMQ4wDAYDVQQHDAVQYXJpczEOMAwGA1UECgwFNldJTkQx
FDASBgNVBAsMC0NBIERpdmlzaW9uMR0wGwYDVQQDDBQ2V0lORCBUZXN0IEVDRFNB
IFJDQTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABAe2MZ214KpDX1qTkcQlq9kV
xFTd87XP8C9n+R/Xw5dzguFzG2tVTuzjJ2vTMTSL7l+Yyzoi/4VkdGeY0e7kJPuj
gc0wgcowRAYIKwYBBQUHAQEEODA2MDQGCCsGAQUFBzABhihodHRwczovL3Bpa2Fj
aHUtY2EuaW90LXJhZC5jb206NDQ0My9vY3NwMEIGA1UdHwQ7MDkwN6A1oDOGMWh0
dHBzOi8vcGlrYWNodS1jYS5pb3QtcmFkLmNvbTo0NDQzL2Rvd25sb2Fkcy9jcmww
```

Download

Pressing the Download Action, download a file containing

1. Required Certificate
2. Intermediate issuer certificate (sub-CA)
3. Root CA certificate

Revoke

Certificate revoking is the process of invalidating a digital certificate before its expiration, ensuring it can no longer be trusted.

Pressing the Revoke Action revoking the certificate and later on adding it to the CRL list

Delete

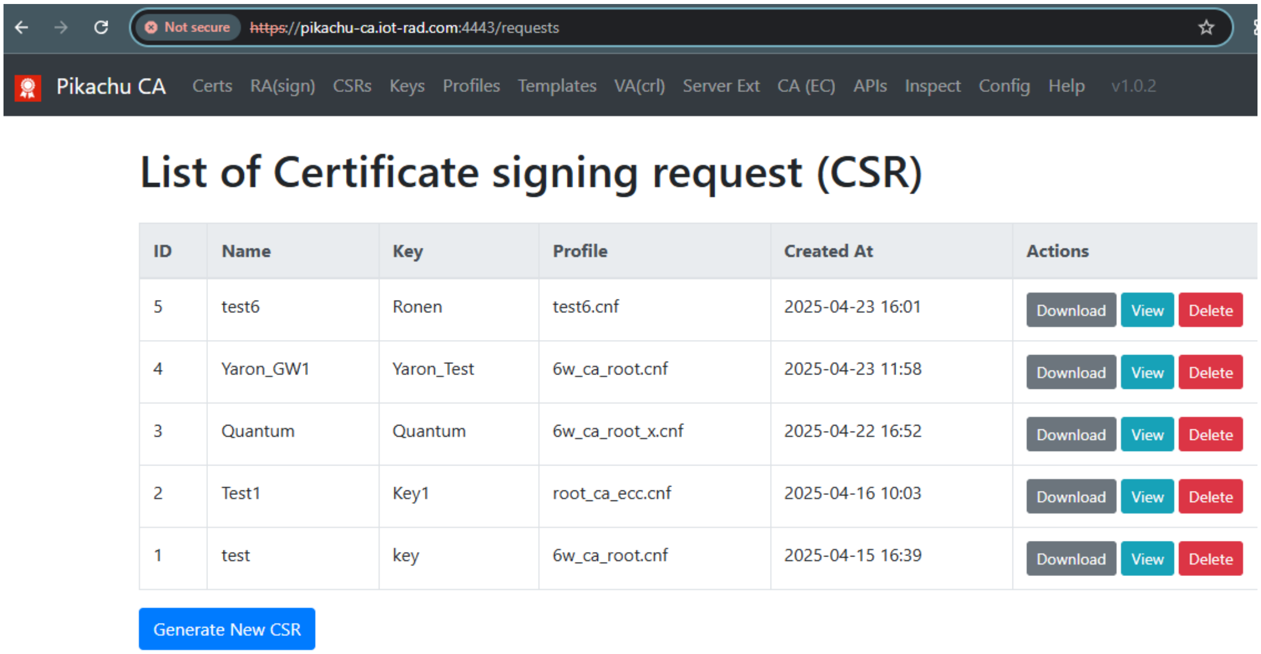
Pressing the Delete Action, Certificate is deleted from DB and form the table

2.10 Certificate Requests

UI

At the main view under management Links select CSR Requests

Redirected to view with list of certificate requests (CSR)



List of Certificate signing request (CSR)

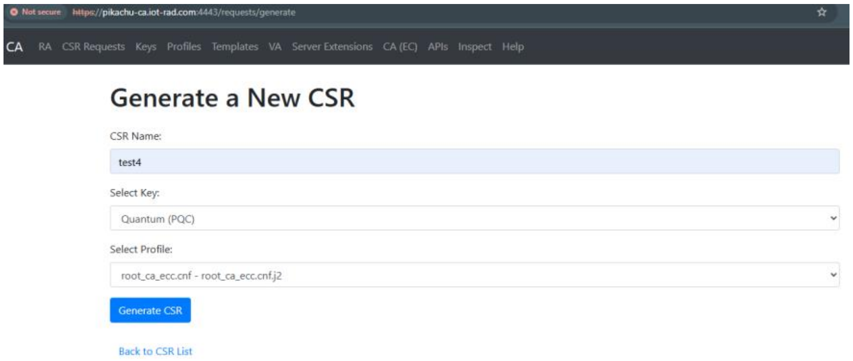
ID	Name	Key	Profile	Created At	Actions
5	test6	Ronen	test6.cnf	2025-04-23 16:01	Download View Delete
4	Yaron_GW1	Yaron_Test	6w_ca_root.cnf	2025-04-23 11:58	Download View Delete
3	Quantum	Quantum	6w_ca_root_x.cnf	2025-04-22 16:52	Download View Delete
2	Test1	Key1	root_ca_ecc.cnf	2025-04-16 10:03	Download View Delete
1	test	key	6w_ca_root.cnf	2025-04-15 16:39	Download View Delete

https://openxpki.iot-rad.com:4443/requests

to generate new CSR press button Generate New CSR

Redirected to Generate a New CSR

In order to make ne CSR select key from list , enter a Name and select a profile



Generate a New CSR

CSR Name: test4

Select Key: Quantum (PQC)

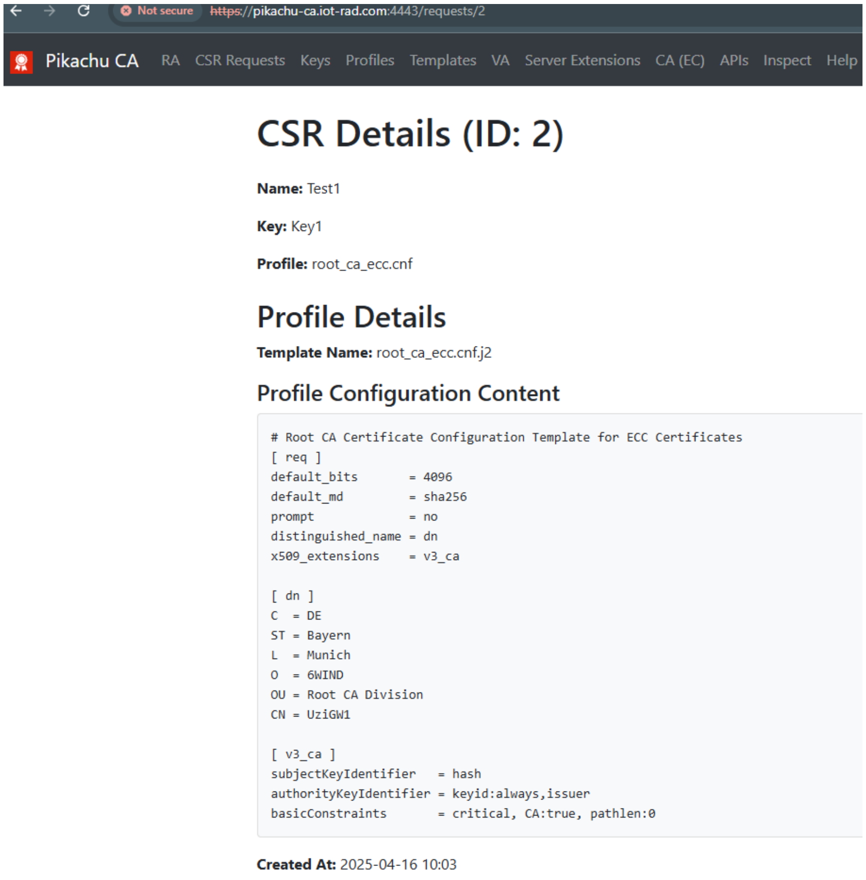
Select Profile: root_ca_ecc.cnf - root_ca_ecc.cnf.j2

https://openxpki.iot-rad.com:4443/requests/generate

press button Generate CSR to generate new one

The view will be redirected to CSR list

Press View at the CSR you wish to see details



CSR Details (ID: 2)

Name: Test1

Key: Key1

Profile: root_ca_ecc.cnf

Profile Details

Template Name: root_ca_ecc.cnf.j2

Profile Configuration Content

```
# Root CA Certificate Configuration Template for ECC Certificates
[ req ]
default_bits       = 4096
default_md         = sha256
prompt             = no
distinguished_name = dn
x509_extensions    = v3_ca

[ dn ]
C  = DE
ST = Bayern
L  = Munich
O  = 6WIND
OU = Root CA Division
CN = UziGW1

[ v3_ca ]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints       = critical, CA:true, pathlen:0
```

Created At: 2025-04-16 10:03

User can copy the PEM certificate and use for any purpose

Linux

Generate Key

```
openssl genrsa -out client1.key 2048
```

Generate request

prepare configuration to be added to the certificate request

for instance,

```
cat > client1.cnf <<EOL
[ req ]
default_bits        = 2048
default_md          = sha256
distinguished_name = req_distinguished_name
attributes          = req_attrs
prompt              = no

[ req_distinguished_name ]
CN                  = client1.example.com
O                   = My Organization
C                   = US

[ req_attrs ]
challengePassword   = SecretChallenge
EOL
```

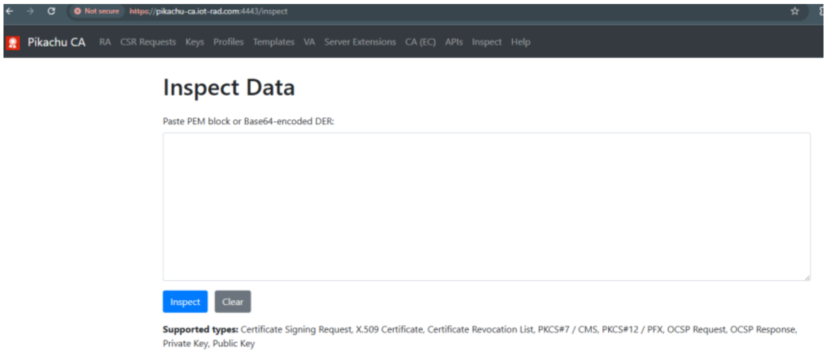
Make the request

```
openssl req -new -key client1.key -config client1.cnf -out client1.csr
```

this request can be used for either UI signing or API based SCEP or EST

2.11 Inspect

User can inspect any PEM block or Base64-encoded DER data



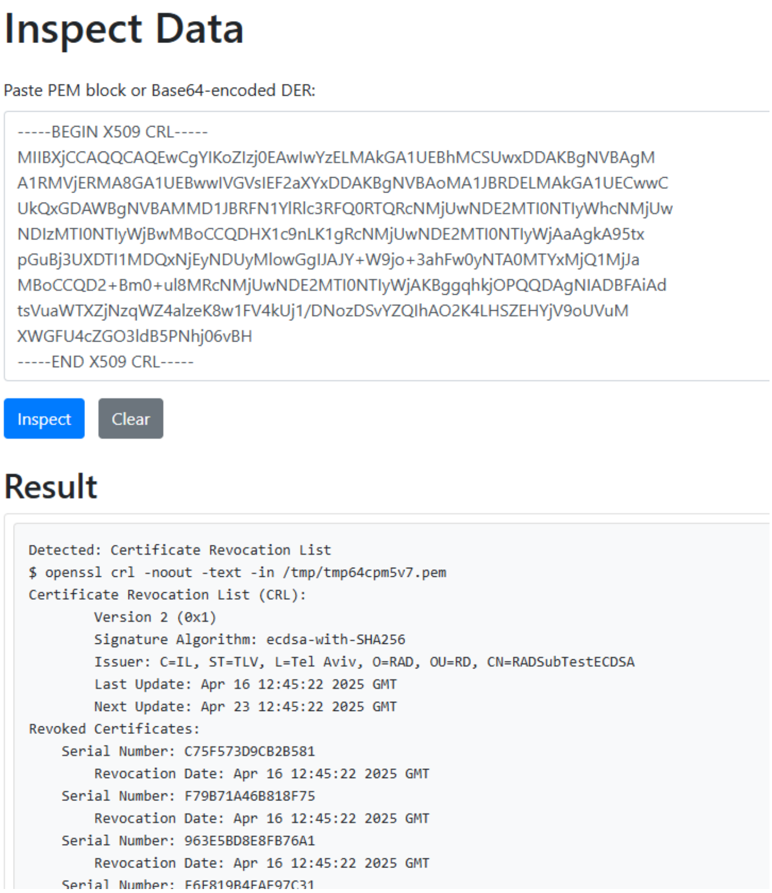
Inspect Data

Paste PEM block or Base64-encoded DER:

Supported types: Certificate Signing Request, X.509 Certificate, Certificate Revocation List, PKCS#7 / CMS, PKCS#12 / PFX, OCSP Request, OCSP Response, Private Key, Public Key

Supported types: Certificate Signing Request, X.509 Certificate, Certificate Revocation List, PKCS#7 / CMS, PKCS#12 / PFX, OCSP Request, OCSP Response, Private Key, Public Key

Inspect provides the designated openssl view stdout



Inspect Data

Paste PEM block or Base64-encoded DER:

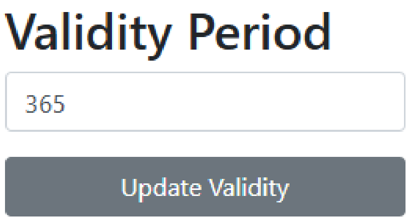
```
-----BEGIN X509 CRL-----
MIIBXjCCAQQCAQEwCgYIKoZIzj0EAwIwYzELMAkGA1UEBhMCSUwxDDAKBgNVBAgM
A1RMVjERMA8GA1UEBwwIVGVsIEF2aXYxDDAKBgNVBAoMA1JBRDELMAkGA1UECwwC
UkQxGDAWBgNVBAMMD1JBRFN1YlRlc3RFQ0RTQRcNMjUwNDE2MTI0NTIyWhcNMjUw
NDIzMTI0NTIyWjBwMBoCCQDHX1c9nLK1gRcNMjUwNDE2MTI0NTIyWjAaAgkA95tx
pGuBj3UXDTI1MDQxNjEyNDUyMlowGgIJAJY+W9jo+3ahFw0yNTA0MTYxMjQ1MjJa
MBoCCQD2+Bm0+ul8MRcNMjUwNDE2MTI0NTIyWjAKBggqhkjOPQQDAgNIADBFAiAd
tsVuaWTXZjNzqWZ4alzeK8w1FV4kUj1/DNozDSvYZQIhAO2K4LHSZEHYjV9oUVuM
XWGFU4cZGO3ldB5PNhj06vBH
-----END X509 CRL-----
```

Result

```
Detected: Certificate Revocation List
$ openssl crl -noout -text -in /tmp/tmp64cpm5v7.pem
Certificate Revocation List (CRL):
        Version 2 (0x1)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C=IL, ST=TLV, L=Tel Aviv, O=RAD, OU=RD, CN=RADSubTestECDSA
        Last Update: Apr 16 12:45:22 2025 GMT
        Next Update: Apr 23 12:45:22 2025 GMT
Revoked Certificates:
    Serial Number: C75F573D9CB2B581
        Revocation Date: Apr 16 12:45:22 2025 GMT
    Serial Number: F79B71A46B818F75
        Revocation Date: Apr 16 12:45:22 2025 GMT
    Serial Number: 963E5BD8E8FB76A1
        Revocation Date: Apr 16 12:45:22 2025 GMT
    Serial Number: F6F819B4FAF97C31
```

2.12 Validity Period

User enrolls Certificate using Validity period (days)



Validity Period

365

Update using “Update Validity” button

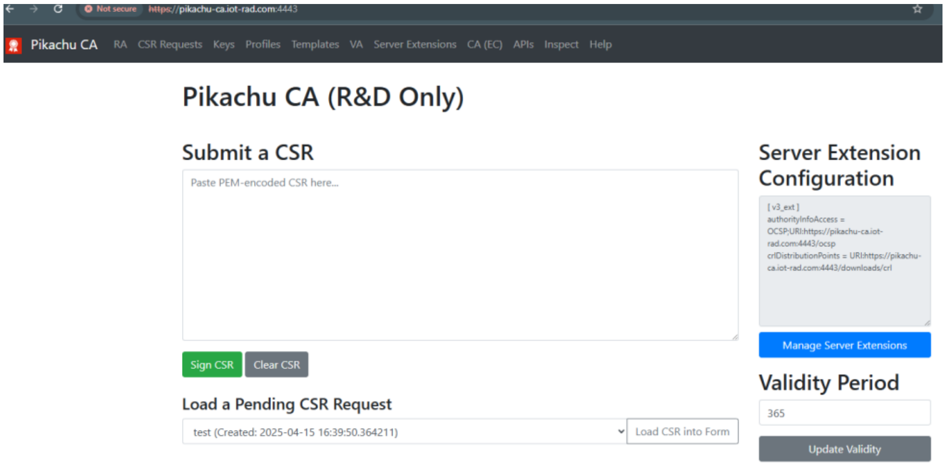
2.13 Server Certificate Extension

Users generate a CSR with custom attributes, but the server may omit, modify, or add attributes before generating the key.

Examples:

1. Adding AIA for OCSP
2. Adding URL for CRL retiring
3. Adding URL for CA chain certificate retiring
4. Omit suggested password given by the used for the enrolment but not needed anymore

Main View contains read only information of the current Server Extension configuration



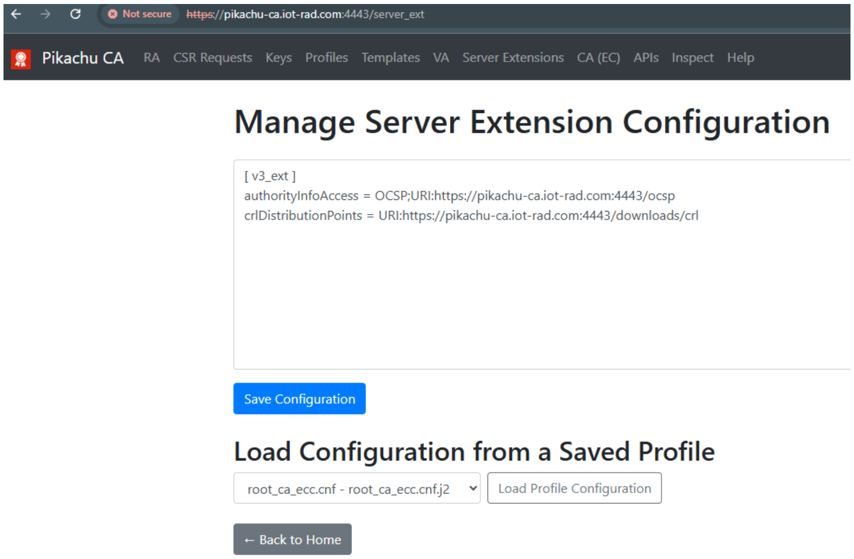
Load a Pending CSR Request: test (Created: 2025-04-15 16:39:50.364211)

Server Extension Configuration:

```
[ v3_ext ]
authorityInfoAccess =
OCSP;URI:https://pikachu-ca.iot-
rad.com:4443/ocsp
crlDistributionPoints = URI:https://pikachu-
ca.iot-rad.com:4443/downloads/crl
```

Validity Period: 365

To Edit the Server extension configuration, press the link “Manage Server Extensions” in a new view Edit and using Save button save the configuration



Manage Server Extension Configuration

```
[ v3_ext ]
authorityInfoAccess = OCSP;URI:https://pikachu-ca.iot-rad.com:4443/ocsp
crlDistributionPoints = URI:https://pikachu-ca.iot-rad.com:4443/downloads/crl
```

Load Configuration from a Saved Profile

root_ca_ecc.cnf - root_ca_ecc.cnf.j2

Using link Back To Home return to main view

Notes :

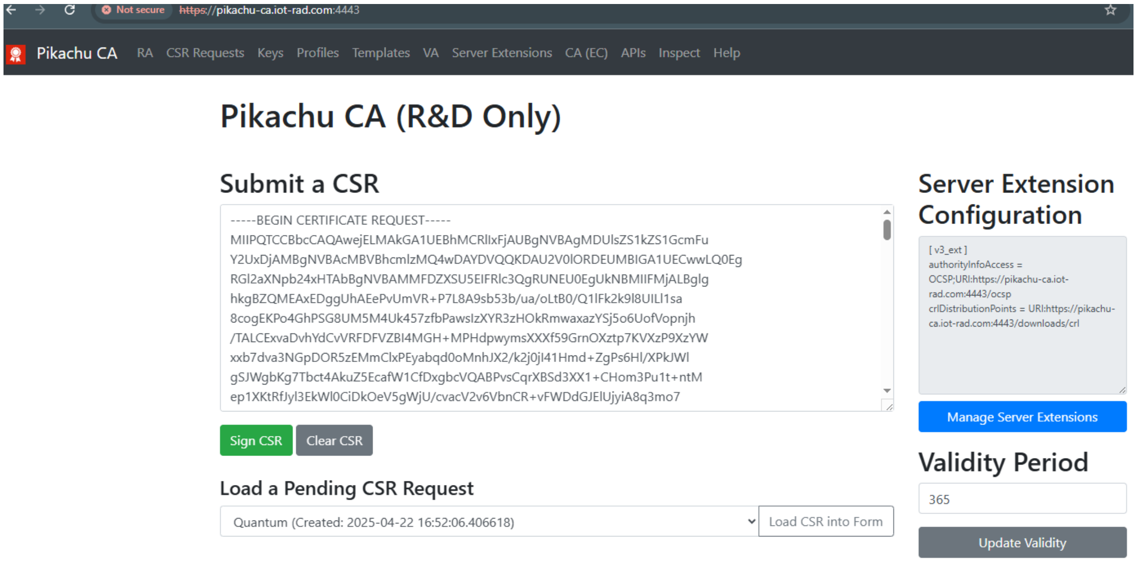
1. only authorityInfoAccess (AIA) and subjectAltName (SAN) supported .

2.14 Enrollment

UI signing

Using the web browser under Submit a CSR past the certificate request content and press Sign CSR button

Users can load existing CSR from the list of Pending CSR Request



Pikachu CA (R&D Only)

Submit a CSR

```
-----BEGIN CERTIFICATE REQUEST-----
MIIPQTCCBbcCAQAwejELMAkGA1UEBhMCRlIxFjAUBgNVBAgMDUlsZS1kZS1GcmFu
Y2UxDjAMBgNVBAcMBVBhcmlzMQ4wDAYDVQQKDAU2V0lORDEUMBIGA1UECwwLQ0Eg
RGl2aXNpb24xHTAbBgNVBAMMFDZXSU5EIFRlc3QgRUNEU0EgUkNBMIIFMjALBglg
hkgBZQMEAxEDggUhAEePvUmVR+P7L8A9sb53b/ua/oLtB0/Q1lFk2k9l8UILl1sa
8cogEKPo4GhPSG8UM5M4Uk457zfbPawsIzXYR3zHOkRmwaxazYSj5o6UofVopnjh
/TALCExvaDvhYdCvVRFDFVZBI4MGH+MPHdpwymsXXXf59GrnOXztp7KVXzP9XzYW
xxb7dva3NGpDOR5zEMmClxPEyabqd0oMnhJX2/k2j0jI41Hmd+ZgPs6Hl/XPkJWl
gSJWgbKg7Tbct4AkuZ5EcafW1CfDxgbcVQABPvsCqrXBSd3XX1+CHom3Pu1t+ntM
ep1XKtRfJyl3EkWl0CiDkOeV5gWjU/cvacV2v6VbnCR+vFWDdGJElUjyiA8q3mo7
```

Sign CSR | Clear CSR

Load a Pending CSR Request

Quantum (Created: 2025-04-22 16:52:06.406618) | Load CSR into Form

Server Extension Configuration

```
[ v3_ext ]
authorityInfoAccess =
OCSP;URI:https://pikachu-ca.iot-
rad.com:4443/ocsp
crlDistributionPoints = URI:https://pikachu-
ca.iot-rad.com:4443/downloads/crl
```

Manage Server Extensions

Validity Period

365

Update Validity

After signing the certificate will appear in the issued certificates table below

The table can be filtered by either part of Common Name or Serial values

EST

EST enrollment technology automates certificate issuance by allowing devices to submit CSRs and receive signed certificates over secure channels.

Convert the certificate request

```
openssl req -outform DER -in client1.csr -out client1.csr.der
```

send the enrolment command using curl

```
curl -k -X POST --data-binary @client1.csr.der \
https://openxpki.iot-rad.com:4443/.well-known/est/simpleenroll \
  -H "Content-Type: application/pkcs10" \
  --output client1.crt.p7
```

Extract the certificate

```
openssl pkcs7 -inform DER -in client1.crt.p7 -print_certs -out client1.crt
```

the certificate added to the UI issues certificates table

tested issued certificate using openssl

```
openssl x509 -in client1.crt  -noout -text
```

enroll using command estclient

```
estclient enroll -server pikachu-ca.iot-rad.com:4443 -insecure -csr etx.csr -
out etx.crt
```

SCEP (in development)

For SCEP you must provide the CA of either full chain (Root + Sub CA) or only the intimidate one (Sub CA)

send the enrolment command using sscep command (see Complementary section for installing it)

```
sscep enroll -u http://openxpki.iot-rad.com:8090/scep \
-c ca.cert.pem \
-k client1.key \
-r client1.csr \
-l client1.crt
```

Note: to debug sscep command add -d option

the certificate added to the UI issues certificates table

tested issued certificate using openssl

```
openssl x509 -in client1.crt  -noout -text
```

Convert the certificate request

```
openssl req -outform DER -in client1.csr -out client1.csr.der
```

send the enrolment command using curl

```
curl -k -X POST --data-binary @client1.csr.der \
https://openxpki.iot-rad.com:4443/.well-known/est/simpleenroll \
  -H "Content-Type: application/pkcs10" \
  --output client1.crt.p7
```

Extract the certificate

```
openssl pkcs7 -inform DER -in client1.crt.p7 -print_certs -out client1.crt
```

the certificate added to the UI issues certificates table

tested issued certificate using openssl

```
openssl x509 -in client1.crt  -noout -text
```

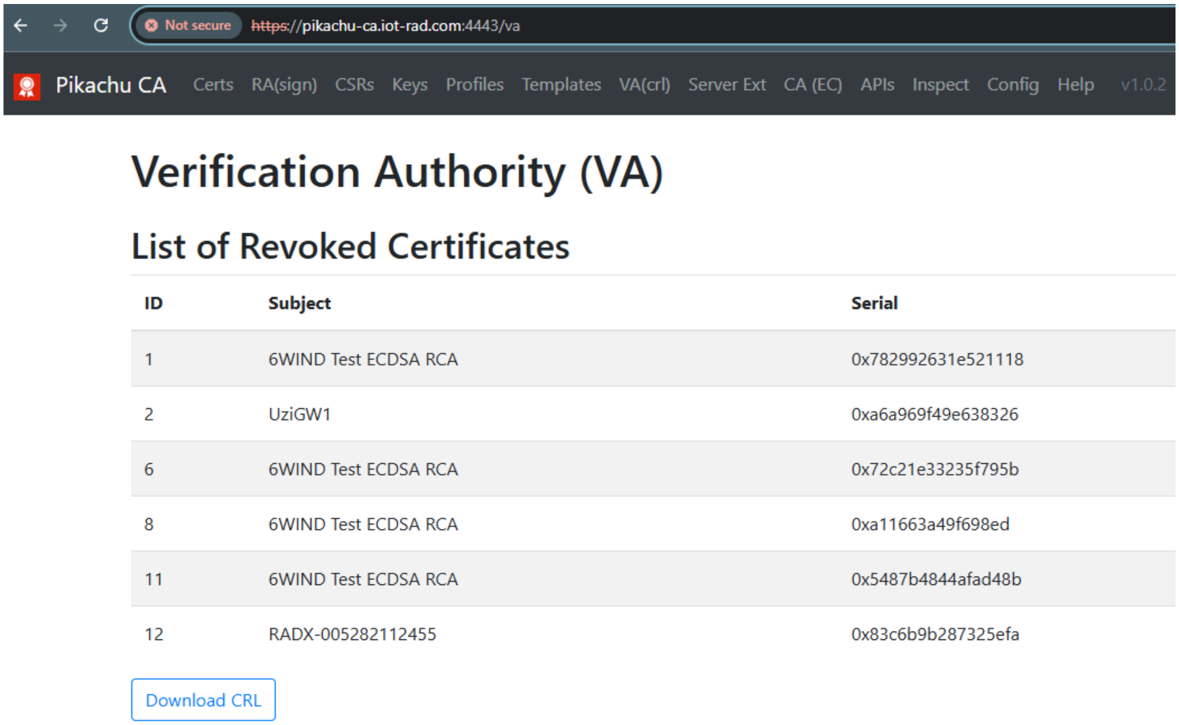
Note: based on GitHub repo https://github.com/mosen/SCEPy

2.15 Verification Authority (VA)

CRL (Certificate Revocation List) is a mechanism for maintaining and distributing a list of digital certificates that have been revoked by a Certificate Authority, ensuring that clients can verify certificate validity.

View

List of revoked certificates



Verification Authority (VA)

List of Revoked Certificates

ID	Subject	Serial
1	6WIND Test ECDSA RCA	0x782992631e521118
2	UziGW1	0xa6a969f49e638326
6	6WIND Test ECDSA RCA	0x72c21e33235f795b
8	6WIND Test ECDSA RCA	0xa11663a49f698ed
11	6WIND Test ECDSA RCA	0x5487b4844afad48b
12	RADX-005282112455	0x83c6b9b287325efa

Download

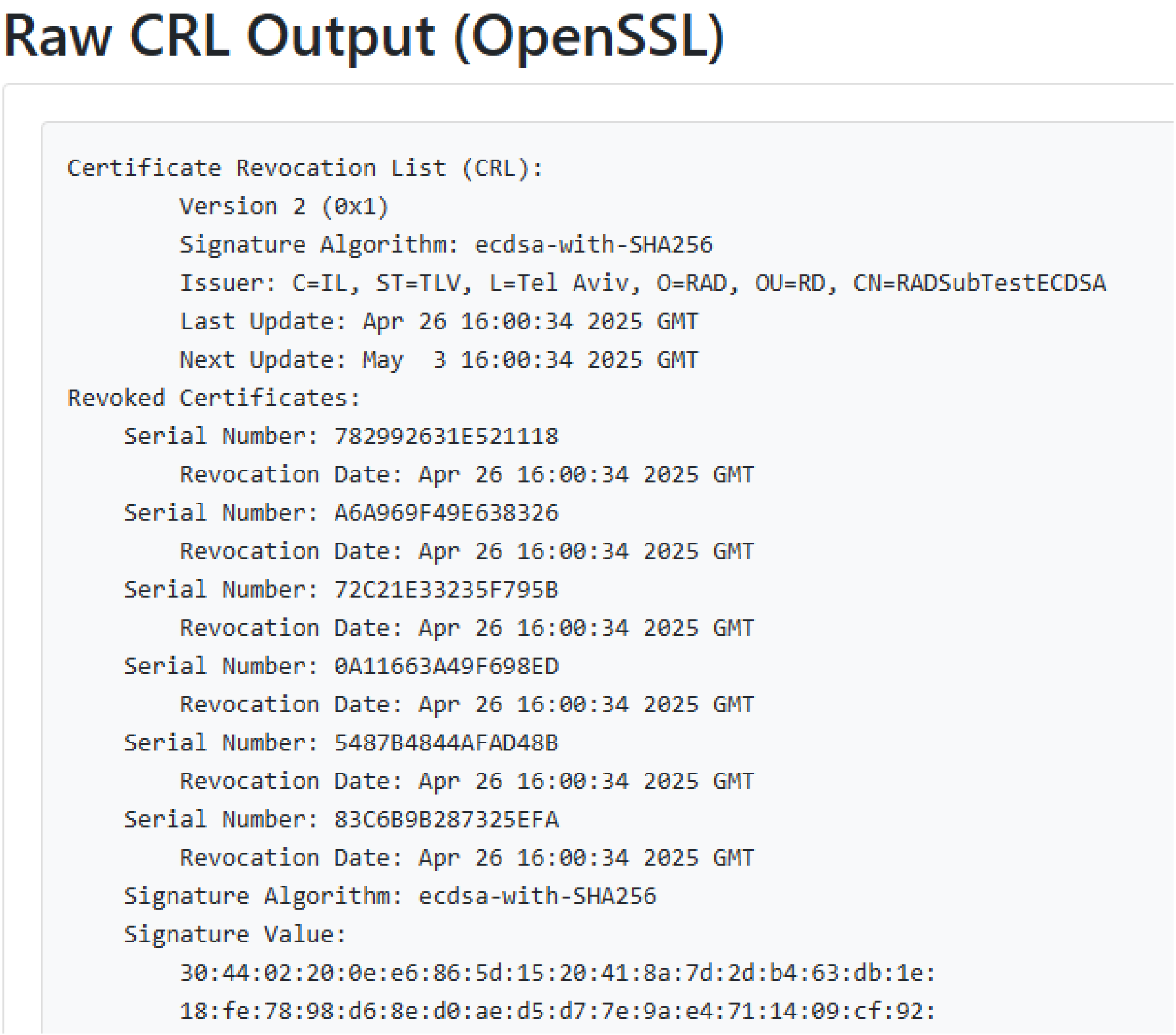
Download the CRL by two methods:

1. Using UI pressing the link Download CRL
2. Using API endpoint

```
curl -k https://openxpki.iot-rad.com:4443/downloads/crl --output crl.pem
```

Inspect

Openssl based info.



Raw CRL Output (OpenSSL)

```
Certificate Revocation List (CRL):
        Version 2 (0x1)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C=IL, ST=TLV, L=Tel Aviv, O=RAD, OU=RD, CN=RADSubTestECDSA
        Last Update: Apr 26 16:00:34 2025 GMT
        Next Update: May  3 16:00:34 2025 GMT
Revoked Certificates:
    Serial Number: 782992631E521118
        Revocation Date: Apr 26 16:00:34 2025 GMT
    Serial Number: A6A969F49E638326
        Revocation Date: Apr 26 16:00:34 2025 GMT
    Serial Number: 72C21E33235F795B
        Revocation Date: Apr 26 16:00:34 2025 GMT
    Serial Number: 0A11663A49F698ED
        Revocation Date: Apr 26 16:00:34 2025 GMT
    Serial Number: 5487B4844AFAD48B
        Revocation Date: Apr 26 16:00:34 2025 GMT
    Serial Number: 83C6B9B287325EFA
        Revocation Date: Apr 26 16:00:34 2025 GMT
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:44:02:20:0e:e6:86:5d:15:20:41:8a:7d:2d:b4:63:db:1e:
        18:fe:78:98:d6:8e:d0:ae:d5:d7:7e:9a:e4:71:14:09:cf:92:
```

MQTTs with CRL file

Common Server using the CRL file to deny access of client with revoked certification is MQTT broker mosquitto.

MQTT added configuration in mosquitto.conf (see Paragraph 2.2 Insulation / complementary /MQTT Broker

```
crlfile /mosquitto/certs/crl_client_2.pem
```

the MQTT server deny revoked certificates based connection attempts

the server must be periodicly updated with new updated CRL

example adding crl updater to the docker compose

```
  crl-updater:
    image: alpine:latest
    container_name: crl_updater
    volumes:
      - ./mosquitto/certs:/mosquitto/
      - /var/run/docker.sock:/var/run/docker.sock  # So we can signal the
mosquitto container
    # Install curl and bash, then run a loop:
    command: >
      sh -c "apk add --no-cache curl bash &&
      while true; do
         echo 'Downloading new CRL file...';
         curl -k https://openxpki.iot-rad.com:4443/downloads/crl --output
/mosquitto/certs/crl.pem;
         echo 'Triggering Mosquitto to reload its configuration...';
         docker kill --signal=SIGHUP mosquitto;
         echo 'Sleeping for 24 hours before next update...';
         sleep 86400;
      done"
    restart: unless-stopped
```

Note: not tested yet

2.16 CA certificate management

User can retrieve the following CA certificate

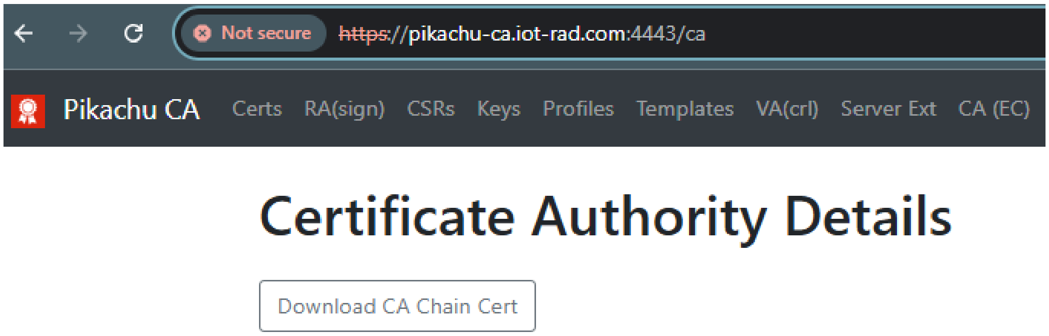
Root CA Certificate:
A trusted self-signed certificate that sits at the top of the certificate hierarchy. It is the anchor of trust used to verify all other certificates in the chain.

Sub CA (Intermediate) Certificate:
A certificate issued by the Root CA (or another intermediate) to delegate signing authority. It acts as a bridge between the Root CA and end-entity (leaf) certificates, improving security and scalability.

Download

Download the CA Chain Certificate by two methods:

1. Using UI pressing the link Download the CA Chain Cert



2. Using API endpoint

```
curl -k https://openxpki.iot-rad.com:4443/downloads/chain --output
ca.chain.pem
```

the ca.chain.pem file can be inspected using openssl

```
openssl x509 -in ca.chain.pem -noout -text
```

View

Pressing the buttons “View Root CA Certificate” and “View Sub CA Certificate” redirect to to view page with all needed Data

Update

Change the configuration file config.ini

```
[CA]
# Which subordinate CA to use by default: "EC" or "RSA"
mode = EC

# Paths for both modes; the get_ca_config() helper below will pick the right
SUBCA_KEY_PATH_EC     = /home/rocky/pki-subca/rad_ca_sub_ec.key
SUBCA_CERT_PATH_EC    = /home/rocky/pki-subca/rad_ca_sub_ec.crt
CHAIN_FILE_PATH_EC    = /home/rocky/pki-subca/rad_chain_ec.crt

SUBCA_KEY_PATH_RSA    = /home/rocky/pki-subca/rad_ca_sub_rsa.key
SUBCA_CERT_PATH_RSA   = /home/rocky/pki-subca/rad_ca_sub_rsa.crt
CHAIN_FILE_PATH_RSA   = /home/rocky/pki-subca/rad_chain_rsa.crt

ROOT_CERT_PATH        = /home/rocky/pki-root/rad_ca_root.crt
[HTTPS]
# HTTPS certificate & key for your main CA UI
ssl_cert = /home/rocky/pki-https/tls.cert.pem
ssl_key  = /home/rocky/pki-https/tls.key.pem
port     = 4443
```

Generate

Below example how to generate all needed CA files

Root CA

Generate Key using type EC and curve prime256v1(secp256r1)

Use root CA request configuration

```
cat > rad_ca_root.cnf <<EOL
# CA Certificate Configuration Template for Root ECC Certificates
[ req ]
# Note: For ECC keys the “default_bits” option is not used.
default_md          = sha256
default_days        = 3650
prompt              = no
distinguished_name = dn
x509_extensions     = v3_ca
default_ec_curve    = prime256v1

[ dn ]
C  = IL
ST = TLV
L  = Tel Aviv
O  = RAD
OU = RD
CN =  RADRootTestECDSA

[ v3_ca ]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints       = critical, CA:true, pathlen:0
keyUsage               = critical, digitalSignature, keyCertSign, cRLSign
EOL
```

Sign the root by itself

```
openssl req -config rad_ca_root.cnf -key rad_ca_root.key -new -x509 -days
3650 -sha256 -out rad_ca_root.crt
```

Sub CA

Generate Key using type EC and curve prime256v1(secp256r1)

Use sub-CA request configuration

```
cat > rad_ca_sub.cnf <<EOL
# CA Certificate Configuration Template for Subordinate (Intermediate) ECC
Certificates
[ req ]
default_bits       = 2048
default_md         = sha256
prompt             = no
distinguished_name = dn
req_extensions     = v3_intermediate

[ dn ]
C  = IL
ST = TLV
L  = Tel Aviv
O  = RAD
OU = RD
CN = RADSubTestECDSA

[ v3_intermediate ]
subjectKeyIdentifier   = hash
#authorityKeyIdentifier = keyid,issuer
basicConstraints       = critical, CA:true, pathlen:0
keyUsage               = keyCertSign, cRLSign
EOL
```

Prepare the Sub-CA Certificate request

```
openssl req -config rad_ca_sub.cnf -key rad_ca_sub.key -new -out
rad_ca_sub.csr
```

add the Root-CA signing Server extension

```
cat > ca_root_ext.cnf <<EOL
[ v3_intermediate ]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid,issuer
basicConstraints       = critical, CA:true, pathlen:0
keyUsage               = digitalSignature,keyCertSign, cRLSign
crlDistributionPoints  = URI:https://openxpki.iot-rad.com:4443/downloads/crl
EOL
```

Sign the sub-CSR by the root CA

```
openssl x509 -req -in rad_ca_sub.csr -CA rad_ca_root.crt -CAkey
rad_ca_root.key -CAcreateserial -out rad_ca_sub.crt -days 3650 -sha256 -
extfile ca_root_ext.cnf -extensions v3_intermediate
```

this provides the certificates and keys for both root and sub CA.

2.17 Online Certificate Status Protocol (OCSP)

OCSP is A real-time certificate status checking protocol that allows clients to verify whether a digital certificate has been revoked, without downloading the full CRL. OCSP improves performance and bandwidth usage compared to traditional revocation lists.

In order to perform the check the user must have the following files:

1. CA certificate (can be only Sub-CA)
2. Certificate

Check certificate status command ca certificate

```
openssl ocsp -reqout ocsp_request.der \
-CAfile ca.chain.pem \
-issuer ca.cert.pem
-cert client1.crt \
-url https://openxpki.iot-rad.com:4443/ocsp \
-resp_text -respout ocsp_response.der
```

The command stdout return the certificate status

Valid certificate result :

```
Response verify OK
/home/rocky/pki-ocsp/cert_2.pem: good
        This Update: Apr  3 13:15:21 2025 GMT
        Next Update: Apr 10 13:15:21 2025 GMT
```

Revokes certificate result :

```
Response verify OK
/home/rocky/pki-ocsp/cert_revoked.pem: revoked
        This Update: Apr  3 13:15:57 2025 GMT
        Next Update: Apr 10 13:15:57 2025 GMT
        Reason: unspecified
        Revocation Time: Apr  3 13:15:57 2025 GMT
```

the output file ocsp_response.der contains the status as well

command

```
openssl ocsp -respin ocsp_response.der -text -noverify
```

stdout

```
OCSP Response Data:
    OCSP Response Status: successful (0x0)
    Response Type: Basic OCSP Response
    Version: 1 (0x0)
    Responder Id: 9DB644062A4D85759C46D1C4215F4DB7C345149D
    Produced At: Apr  3 10:58:10 2025 GMT
    Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 616FC051FA23823B80B63EDD49EBFE18F4FB4E77
      Issuer Key Hash: 9DB644062A4D85759C46D1C4215F4DB7C345149D
      Serial Number: 0DCD62C3C11C144DD8AA313FF654067D3F413E5F
    Cert Status: revoked
    Revocation Time: Apr  3 10:58:10 2025 GMT
    Revocation Reason: unspecified (0x0)
    This Update: Apr  3 10:58:10 2025 GMT
    Next Update: Apr 10 10:58:10 2025 GMT
```

Certificate based Servers using OCSP

In order for TLS based Servers to use OCSP the information of the URL to check must be embedded within the Certificate itself.

OCSP requires authorityInfoAccess (AIA)

Adding attributes by following paragraph 2.5 Server Certificate Extension

Some TLS servers like HAProxy only have mode configuration attribute the URL is taken from the certificate attributes.

2.18 Post-Quantum Keys

Users can issue certificate using stronger Keys algorithms

Prerequisite to do so is to add new provider (extension to openssl command)

Check if Quantum safe keys extension oqsprovider Is activated using the following command:

```
#run command
openssl list -providers
#output
Providers:
  default
    name: OpenSSL Default Provider
    version: 3.2.2
    status: active
  oqsprovider
    name: OpenSSL OQS Provider
    version: 0.8.1-dev
    status: active
```

Generate key

```
openssl genpkey -algorithm mldsa44 -provider oqsprovider -out client1.key
```

checking all possible algorithms for the oqsprovider provider with the command:

```
openssl list -public-key-algorithms -provider oqsprovider
```

Generate request

prepare configuration to be added to the certificate request

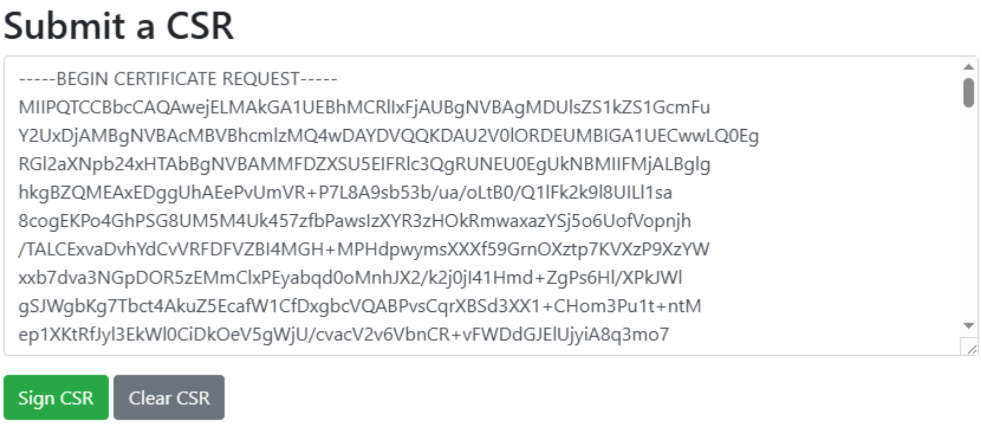
for instance,

```
cat > client1.cnf <<EOL
[ req ]
default_bits        = 2048
default_md          = sha256
distinguished_name = req_distinguished_name
attributes          = req_attrs
prompt              = no

[ req_distinguished_name ]
CN                  = client1.example.com
O                   = My Organization
C                   = US

[ req_attrs ]
challengePassword  = SecretChallenge
EOL
```

Make the request

```
openssl req -new -key client1.key -config client1.cnf -out client1.csr
```

using UI

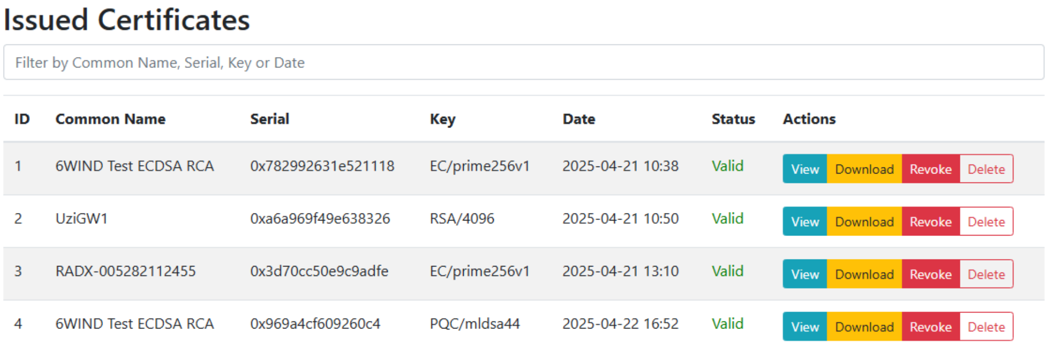


Submit a CSR

```
-----BEGIN CERTIFICATE REQUEST-----
MIIPQTCCBbcCAQAwejELMAkGA1UEBhMCRlIxFjAUBgNVBAgMDUlsZS1kZS1GcmFu
Y2UxDjAMBgNVBAcMBVBhcmlzMQ4wDAYDVQQKDAU2V0lORDEUMBIGA1UECwwLQ0Eg
RGl2aXNpb24xHTAbBgNVBAMMFDZXSU5EIFRlc3QgRUNEU0EgUkNBMIIFMjALBglg
hkgBZQMEAxEDggUhAEePvUmVR+P7L8A9sb53b/ua/oLtB0/Q1lFk2k9l8UILl1sa
8cogEKPo4GhPSG8UM5M4Uk457zfbPawsIzXYR3zHOkRmwaxazYSj5o6UofVopnjh
/TALCExvaDvhYdCvVRFDFVZBI4MGH+MPHdpwymsXXXf59GrnOXztp7KVXzP9XzYW
xxb7dva3NGpDOR5zEMmClxPEyabqd0oMnhJX2/k2j0jI41Hmd+ZgPs6Hl/XPkJWl
gSJWgbKg7Tbct4AkuZ5EcafW1CfDxgbcVQABPvsCqrXBSd3XX1+CHom3Pu1t+ntM
ep1XKtRfJyl3EkWl0CiDkOeV5gWjU/cvacV2v6VbnCR+vFWDdGJElUjyiA8q3mo7
```

Note: the 2.5 Enrollment using UI can't be used for this operation, code is based on python libraries whereas quantum safe signing is based on OS openssl in the python code.

After signing   the certificate will appear in the issued certificates table bellow



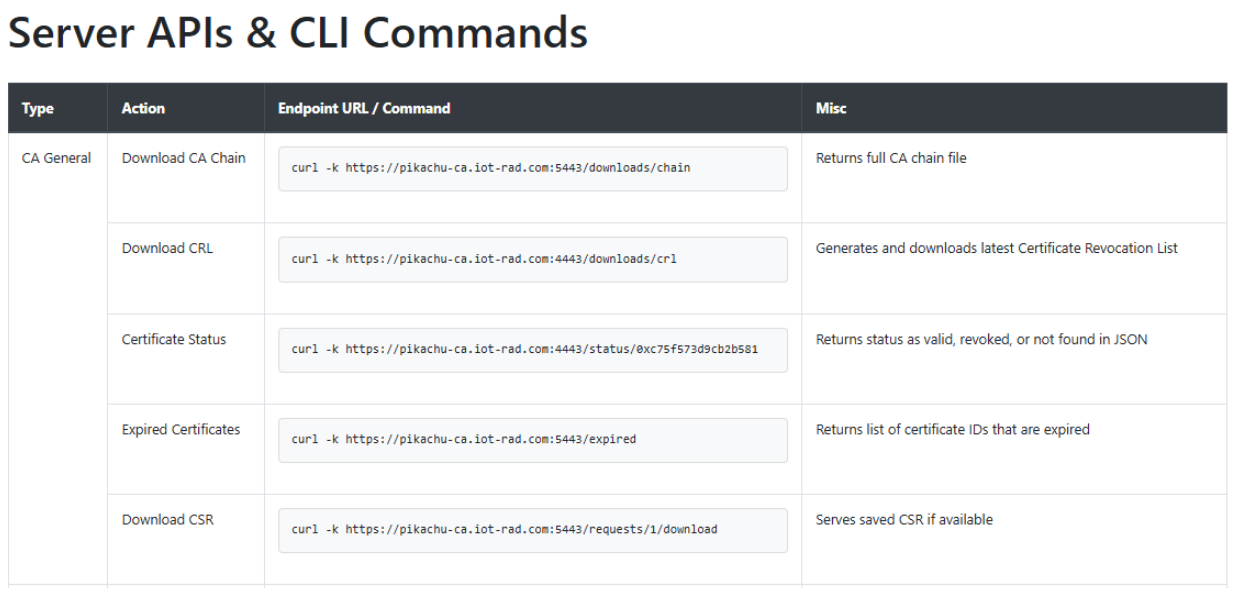
Issued Certificates

ID	Common Name	Serial	Key	Date	Status	Actions
1	6WIND Test ECDSA RCA	0x782992631e521118	EC/prime256v1	2025-04-21 10:38	Valid	View Download Revoke Delete
2	UziGW1	0xa6a969f49e638326	RSA/4096	2025-04-21 10:50	Valid	View Download Revoke Delete
3	RADX-005282112455	0x3d70cc50e9c9adfe	EC/prime256v1	2025-04-21 13:10	Valid	View Download Revoke Delete
4	6WIND Test ECDSA RCA	0x969a4cf609260c4	PQC/mldsa44	2025-04-22 16:52	Valid	View Download Revoke Delete

2.19 APIs

List of supported server APIs

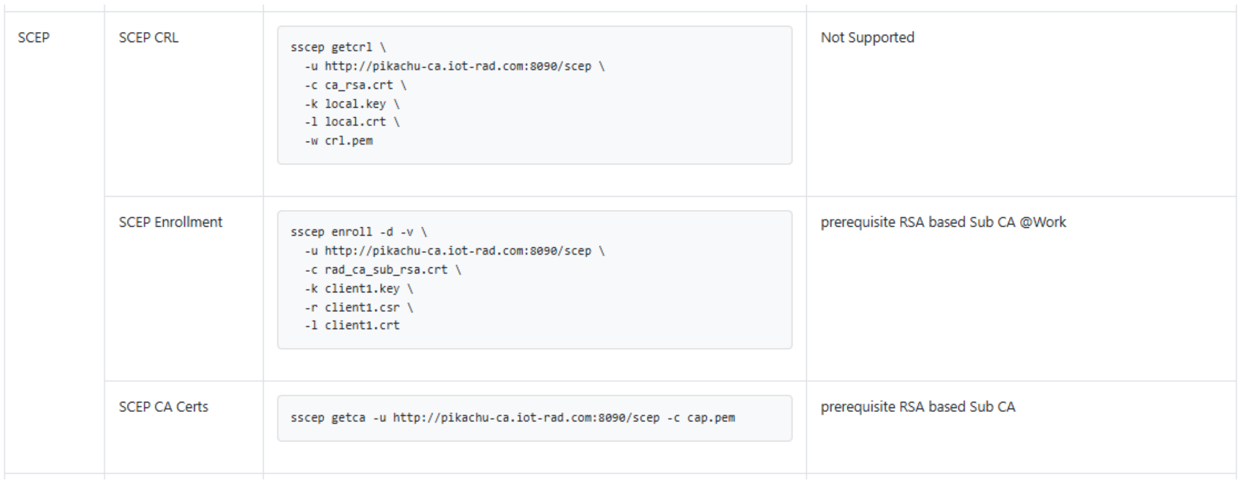
General Server APIs



Server APIs & CLI Commands

Type	Action	Endpoint URL / Command	Misc
CA General	Download CA Chain	`curl -k https://pikachu-ca.iot-rad.com:5443/downloads/chain`	Returns full CA chain file
	Download CRL	`curl -k https://pikachu-ca.iot-rad.com:4443/downloads/crl`	Generates and downloads latest Certificate Revocation List
	Certificate Status	`curl -k https://pikachu-ca.iot-rad.com:4443/status/0xc75f573d9cb2b581`	Returns status as valid, revoked, or not found in JSON
	Expired Certificates	`curl -k https://pikachu-ca.iot-rad.com:5443/expired`	Returns list of certificate IDs that are expired
	Download CSR	`curl -k https://pikachu-ca.iot-rad.com:5443/requests/1/download`	Serves saved CSR if available

SCEP supported APIs

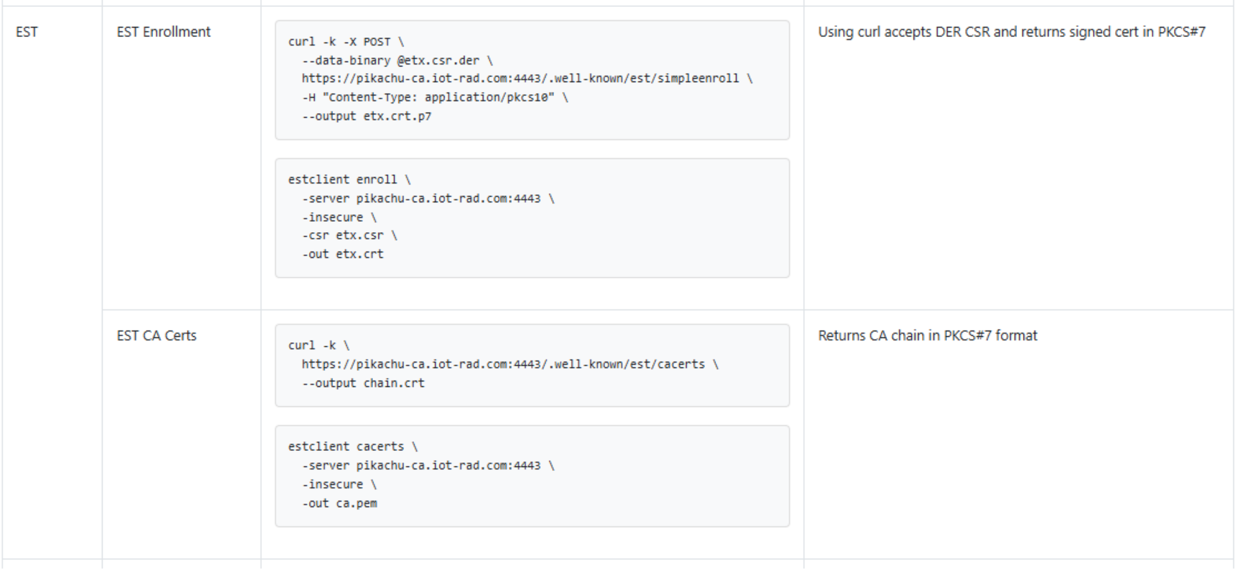


SCEP	SCEP CRL	`sscep getcrl \`
		`  -u http://pikachu-ca.iot-rad.com:8090/scep \`
		`  -c ca_rsa.crt \`
		`  -k local.key \`
		`  -l local.crt \`
		`  -w crl.pem`
			Not Supported

	SCEP Enrollment	`sscep enroll -d -v \`
		`  -u http://pikachu-ca.iot-rad.com:8090/scep \`
		`  -c rad_ca_sub_rsa.crt \`
		`  -k client1.key \`
		`  -r client1.csr \`
		`  -l client1.crt`
			prerequisite RSA based Sub CA @Work

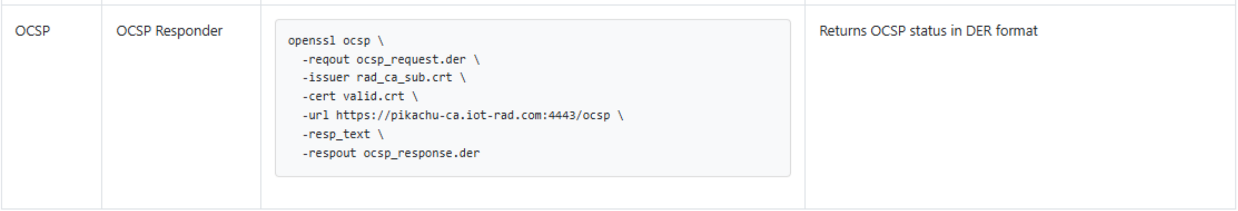
	SCEP CA Certs	`sscep getca -u http://pikachu-ca.iot-rad.com:8090/scep -c cap.pem`	prerequisite RSA based Sub CA

EST supporting Apis (curl and estclient)



EST	EST Enrollment	curl -k -X POST \\ --data-binary @etx.csr.der \\ https://pikachu-ca.iot-rad.com:4443/.well-known/est/simpleenroll \\ -H "Content-Type: application/pkcs10" \\ --output etx.crt.p7 estclient enroll \\ -server pikachu-ca.iot-rad.com:4443 \\ -insecure \\ -csr etx.csr \\ -out etx.crt	Using curl accepts DER CSR and returns signed cert in PKCS#7
	EST CA Certs	curl -k \\ https://pikachu-ca.iot-rad.com:4443/.well-known/est/cacerts \\ --output chain.crt estclient cacerts \\ -server pikachu-ca.iot-rad.com:4443 \\ -insecure \\ -out ca.pem	Returns CA chain in PKCS#7 format

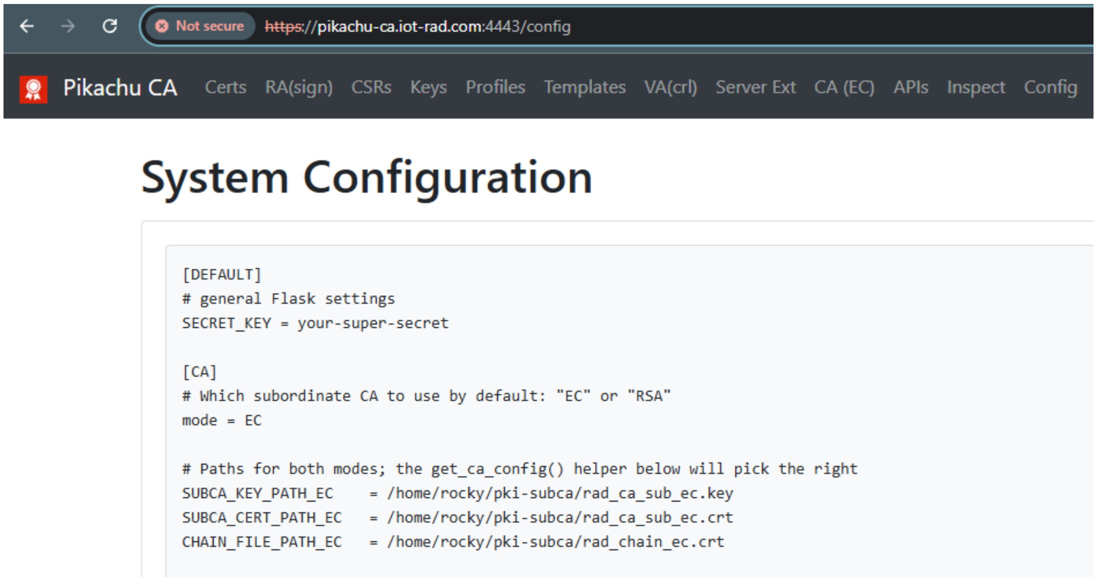
OCSP supporting API



OCSP	OCSP Responder	openssl ocsp \\ -reqout ocsp_request.der \\ -issuer rad_ca_sub.crt \\ -cert valid.crt \\ -url https://pikachu-ca.iot-rad.com:4443/ocsp \\ -resp_text \\ -respout ocsp_response.der	Returns OCSP status in DER format

2.20 Config

Config Navigation Tab shows the server current configuration



System Configuration

```
[DEFAULT]
# general Flask settings
SECRET_KEY = your-super-secret

[CA]
# Which subordinate CA to use by default: "EC" or "RSA"
mode = EC

# Paths for both modes; the get_ca_config() helper below will pick the right
SUBCA_KEY_PATH_EC    = /home/rocky/pki-subca/rad_ca_sub_ec.key
SUBCA_CERT_PATH_EC   = /home/rocky/pki-subca/rad_ca_sub_ec.crt
CHAIN_FILE_PATH_EC   = /home/rocky/pki-subca/rad_chain_ec.crt
```

2.21 Help

PDF format of this word document

Change History

Date of Issue	Revisi on	Author	Responsible	Change Summary	Page Count
8-Apr-2025	1.00	Uzi G.	Asaf B.	Document baseline	42
16-Apr-2025	1.01	Uzi G.	Asaf B.	Update UI, CA details, Validity time manage	43
23-Apr-2025	1.02	Uzi G.	Asaf B.	Update UI , New Navigate tabs. configuration based	52
18-Jun-2025	1,03	Uzi G.		Add manufacture certificate URL end point, Add generation of root and sub certificate procedure	57