



Your Network's Edge

Company Confidential

Pikachu-CA PKI Hands-On 2.0.0

Last updated	4-Dec-2025
Doc. version	2.0.0
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	5
1.1	Glossary	5
1.2	PKI Concept	5
1.3	Overview	7
	Features.....	7
	Quantum safe keys	10
1.4	Purpose	11
1.5	Scope	11
2	PKI platform	12
2.1	Server OS requirements.....	12
2.2	Installation	12
	Python	12
	Server	14
	Complementary	17
2.3	Generate Root and intermediate certification	20
	Generate Root Certificate	21
	Generating Intermediate (Sub) Certificate.....	22
2.4	Configuration	25
2.5	Run Server.....	30
	Shell Command.....	30
	OS Service.....	30
2.6	Ports.....	32
3	Features.....	34
3.1	Top Navigation Layout.....	34
3.2	Keys Management.....	36
	List of Keys.....	36
	View Key.....	37
	Generate a New Key	39
3.3	Certificate Templates	42
	Certificate Templates list.....	42
	Generate New Certificate Template	44
	View Certificate Template	46
	Edit Certificate Template	47
3.4	Enrollment	47
	Enrollment Policies List	48



Generate Enrollment Policy	49
Edit Enrollment Policy	51
Challenge Passwords.....	52
3.5 Events (System)	54
3.6 Templates (admin role only)	57
Loading/Editing.....	57
View Rendered Certificate Template	59
Creating New Template	59
3.7 Certificates	60
List.....	60
View	63
3.8 CSRs	66
List.....	66
View	68
Generate new CSR	70
3.9 Enrollment	72
UI signing.....	72
EST.....	73
SCEP	74
3.10 Verification Authority (VA).....	76
The Verification Authority page displays all certificates that have been revoked by the CA and provides access to the generated Certificate Revocation List (CRL).	76
List of Revoked Certificates	76
3.11 Inspect	78
3.12 CA certificate management	80
Download	81
View	81
Update.....	82
Generate	82
3.13 Online Certificate Status Protocol (OCSP).....	85
Certificate based Servers using OCSP	86
3.14 Post-Quantum Keys	87
3.15 APIs	89
3.16 Inspect	92
3.17 Config.....	93
3.18 ChatPikachu	93
3.19 Help	94
3.20 Logs.....	95
3.21 Account.....	96
3.22 Users.....	97



Appendixes	105
A.1. Complimentary openssl commands	105
Generate CSR.....	105
A.2. MQTTs.....	106
A.3. APIs	107
 Change History.....	 110

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.

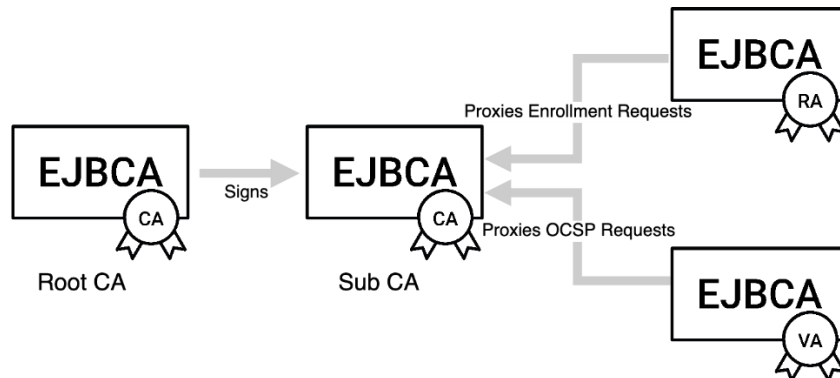
Certificate Revoking - Certificate revoking is the process of invalidating a digital certificate before its expiration, ensuring it can no longer be trusted.

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

Matrix

Feature Category	Capability	Supported	Notes
CA Operations	Issue / revoke / delete certificates	✓	CRL + OCSP supported
	CSR signing	✓	Manual & automated
	Multi-CA mode (RSA/EC)	✓	Selectable via config.ini
	Post-Quantum Keys	✓	Dilithium PQC
Enrollment	SCEP	✓	Uses file-based keys only
	EST	✓	HTTPS + mTLS, Vault optional
	Manual RA signing	✓	Through Web UI
Identity & Access	Role-based access control	✓	Admin/User
	LDAP authentication	✓	Optional
	User state management	✓	Pending/Active/Suspended
Security	Vault private key isolation	✓	Optional, recommended
	Zero-trust key handling	✓	Keys never leave Vault
	Audit logs (server)	✓	Events for all operations
	Vault audit logs	✓	If enabled
Protocols	OCSP	✓	Real-time validation



Feature Category	Capability	Supported	Notes
MQTT	CRL	✓	Download + MQTT enforcement
	TLS Client Auth	✓	Works with Mosquitto
	PowerShell automation scripts	✓	Setup, cleanup, migration
Usability	Health checks	✓	Vault fallback logic
	Web UI	✓	Fully responsive
	REST API	✓	All operations exposed
	Data inspection tool	✓	PEM/DER parsing
	Templates & profiles	✓	Jinja-based

List

Web Interface & API

- Modern Web UI for managing all PKI operations.
- Full RESTful API for automation and integration.

Certificate Lifecycle Management

- Generate keys: **RSA**, **ECC**, and **Post-Quantum (Dilithium)**.
- Create, upload, and sign CSRs.
- View, revoke, and delete issued certificates.
- Download CA certificates, chains, and CRLs.
- Real-time certificate checks via **OCSP**.

Automated Device Enrollment

- **SCEP** protocol for automated enrollment (local keys only).
- **EST** protocol with support for HTTPS and mTLS (Vault optional).
- MQTT TLS authentication with CRL-based revocation enforcement.



Access Control & Multi-Tenancy

- Role-based access control (User/Admin).
- User isolation and managed user states (pending, active, suspended).

Post-Quantum Cryptography

- Built-in support for Dilithium-based PQC keys and CSRs.

HashiCorp Vault Integration (Optional)

- CA private keys stored securely in Vault (never on disk).
- Signing operations performed via Vault PKI engine.
- Separate Vault engines for RSA and EC CA modes.
- Automatic fallback to legacy file-based keys if Vault is unreachable.
- Supports AppRole authentication and custom Vault policies.
- Vault used for Web UI and EST enrollment (*SCEP always uses local keys*).

Security & Architecture

- Strong key isolation when using Vault.
- Secure signing workflows that avoid exposing key material.
- Granular role-based authorization for all CA operations.
- Full audit trail (local events + optional Vault audit logs).
- Zero-trust key handling in Vault mode.
- Automatic failover to local keys for high availability.

Management & Operations

- PowerShell automation scripts for:
 - Starting/stopping the server and Vault
 - Clearing logs, cache, and temp files
 - Setting Vault credentials
 - Database initialization/migration
- Centralized configuration via **config.ini**.

- Adjustable logging (levels & file path).
- Vault health checks and fallback logic.
- Support for Vault-based key rotation.

Protocols & Integrations

- **SCEP**: Automated certificate enrollment.
- **EST**: Automated enrollment with HTTPS/mTLS + Vault support.
- **OCSP**: Online certificate validation.
- **CRL**: Automatic generation and download.
- **LDAP**: Optional directory integration for authentication.

Usability & UI Features

- Filterable and paginated tables for certificates, users, events, keys, etc.
- Download options for certificates, chains, CRLs, PFX, and keys.
- User/admin dashboards with live updates.
- Extensive built-in documentation and quick-start scripts.

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

@'
Flask==3.1.2
Flask-SQLAlchemy==3.1.1
Flask-Login==0.6.3
cryptography==46.0.3
asn1crypto==1.5.1
Werkzeug==3.1.4
SQLAlchemy==2.0.44
Jinja2==3.1.6
MarkupSafe==3.0.3
itsdangerous==2.2.0
click==8.3.1
blinker==1.9.0
cffi==2.0.0
pycparser==2.23
greenlet==3.2.4
typing_extensions==4.15.0
colorama==0.4.6
oscrypto==1.3.0
tzdata==2025.2
hvac==2.1.0
certifi==2025.11.12
charset-normalizer==3.4.4
idna==3.11
requests==2.32.5
urllib3==2.5.0
'@ | Set-Content requirements.txt;

pip3 install -r requirements.txt
```

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



```
|— app.log
|— app.py
|— asnl.py
|— builders.py
|— ca_mode.conf
|— ca.py
|— certificates
|— certs.db
|— chain.crt
|— chainx.crt
|— config_bp.py
|— config.ini
|— config_storage.py
|— db
|   |— certs.db
|— dbtypes.py
|— dummy.key
|— dumps
|— enums.py
|— envelope.py
|— est_cert_chain.p7
|— est_chain.p7
|— est_signed_cert.pem
|— extensions.py
|— FETCH_HEAD
|— html_templates
|   |— api.html
|   |— app.log
|   |— ca.html
|   |— _certificate_detail.html
|   |— config.html
|   |— edit_profile.html
|   |— generate_csr.html
|   |— generate_key.html
|   |— index.html
|   |— inspect.html
|   |— layout.html
|   |— list_certificates.html
|   |— list_certificates.html.171125
|   |— list_certificates.html.delete
|   |— list_csrs.html
|   |— list_keys.html
|   |— list_profiles.html
|   |— list_rendered.html
|   |— list_templates.html
|   |— logs.html
|   |— profile_file.html
|   |— profile_result.html
|   |— rendered_file.html
|   |— rendered_template.html
|   |— server_ext.html
|   |— sign.html
|   |— template_form.html
|   |— va.html
```



```
|— view_csr.html
|— view.html
|— view_key.html
|— logs
|— server.log
|— server.log.1
|— pki-https
|— pikachu_issued_https.crt
|— pikachu_issued_https.key
|— tls.cert.pem
|— tls.key.pem
|— pki-misc
|— crt.pem
|— server_ext.cnf
|— validity.conf
|— pki-root
|— rad_ca_root.crt
|— pki-server-2.code-workspace
|— pki-subca
|— rad_ca_sub_ec.crt
|— rad_ca_sub_ec.key
|— rad_ca_sub_rsa.crt
|— rad_ca_sub_rsa.key
|— rad_chain_ec.crt
|— rad_chain_rsa.crt
|— __pycache__
|— README.md
|— static
|— favicon-16x16.png
|— favicon-32x32.png
|— help.pdf -> PKI-Hands-On1.03.pdf
|— PKI-Hands-On1.02.pdf
|— PKI-Hands-On1.03.pdf
|— version.txt
|— x509_keys.py
|— x509_profiles
|— 6w_ca_root_2.cnf
|— 6w_ca_root_2x.cnf
|— 6w_ca_root.cnf
|— 6w_ca_root_x.cnf
|— bitbucket_server.cnf
|— x509_profiles.py
|— x509_requests.py
|— x509_templates
|— 6w_ca_root_2.cnf.j2
|— 6w_ca_root.cnf.j2
|— 6w_ca_root_ext.cnf.j2
|— 6w_ca_sub.cnf.j2
|— crt_ocsp_ext_alt.cnf.j2
|— crt_ocsp_ext.cnf.j2
|— eon_server_ext.cnf.j2
```


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 compiled, 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_rsa.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

```
openssl 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)

```
PROJ_DIR=~/.pki-server-2
Cp rad_ca_root.key rad_ca_root.crt $PROJ_DIR/pki-root/
cp rad_ca_sub_ec.crt rad_ca_sub_rsa.crt rad_ca_sub_ec.key rad_ca_sub_rsa.key
$PROJ_DIR/pki-subca

cd $PROJ_DIR

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.crtv
```


2.4 Configuration

The Pikachu CA server is configured via the config.ini file.

This file controls CA selection, network ports, logging, LDAP/Vault integration, SCEP behaviour, and all path locations.

[DEFAULT]

- **SECRET_KEY** – Flask secret key used for session security.
- **http_port** – HTTP port used for unauthenticated OCSP responses (default: 80).
- **max_idle_time** – Maximum user idle time before automatic logout (for example: 10h, 4d, 20m).

[LOGGING]

- **log_level** – Log verbosity: TRACE, DEBUG, INFO, WARNING, ERROR, or CRITICAL.
- **log_file** – Path to the main server log file.

[CA]

Controls which sub-CA is used and where CA material is stored.

- **mode** – Default subordinate CA to use: EC or RSA.
- **SUBCA_KEY_PATH_EC / SUBCA_CERT_PATH_EC / CHAIN_FILE_PATH_EC** – Paths for EC sub-CA private key, certificate, and chain file.
- **SUBCA_KEY_PATH_RSA / SUBCA_CERT_PATH_RSA / CHAIN_FILE_PATH_RSA** – Paths for RSA sub-CA private key, certificate, and chain file.
- **ROOT_CERT_PATH** – Path to the root CA certificate.

[VAULT]

Optional HashiCorp Vault integration for key and signing operations.

- **enabled** – Enable or disable Vault (true / false).
- **address** – Vault server URL (for example http://127.0.0.1:8200).
- **role_id, secret_id** – AppRole credentials (typically provided via environment variables).
- **pki_rsa_path, pki_ec_path** – Vault PKI mount paths for RSA and EC.
- **transit_path** – Vault transit engine path for custom signing.
- **timeout** – Connection timeout (seconds).

- **retry_attempts** – Number of retries on Vault connection failure.
- **verify_ssl** – Whether to verify Vault's TLS certificate.
- **ca_cert_path** – CA certificate used to verify Vault when **verify_ssl** is enabled.
- **role_scep**, **role_est**, **role_default** – Vault roles used for SCEP, EST, and general operations.

[LDAP]

Optional LDAP integration for user management.

- **LDAP_HOST / LDAP_PORT** – LDAP server address and port.
- **BASE_DN** – Base DN for LDAP searches.
- **PEOPLE_DN** – DN under which user entries are stored.
- **ADMIN_DN / ADMIN_PASSWORD** – Bind DN and password for the LDAP admin account.
- **enabled** – Enable or disable LDAP authentication (true / false).

[SCEP]

Controls the Simple Certificate Enrollment Protocol behaviour.

- **enabled** – Enable or disable SCEP support.
- **serial_file** – File used to persist the SCEP serial counter across restarts.
- **dump_dir** – Directory where raw SCEP requests are stored (optional).
- **challenge_password_enabled** – Enables challenge-password protection for SCEP enrollment.
- **challenge_password_validity** – Validity period for generated challenge passwords (for example 60m, 2m).

[HTTPS]

Configuration for the main HTTPS interface (UI and APIs).

- **ssl_cert** – Path to the HTTPS server certificate.
- **ssl_key** – Path to the HTTPS private key.
- **port** – HTTPS listen port (default 443).

[TRUSTED_HTTPS]

Configuration for the mTLS / “trusted” HTTPS port.

- **trusted_ssl_cert** – Certificate used for the trusted HTTPS endpoint.
- **trusted_ssl_key** – Private key for the trusted HTTPS endpoint.



- **trusted_port** – Trusted HTTPS port (default 4443).

[PATHS]

Filesystem locations for core PKI artifacts.

- **crl_path** – Path to the CRL file generated by the CA.
- **server_ext_cfg** – Path to the server extension configuration file.
- **validity_conf** – Path to the validity configuration file (default certificate lifetimes).
- **db_path** – Path to the SQLite database (certs.db).



```
[DEFAULT]
# general Flask settings
SECRET_KEY = *****
# HTTP port for unauthenticated OCSP
http_port = 80
# Maximum idle time before automatic logout (e.g., "10h" or "4d" or "20m")
max_idle_time = 30m

[LOGGING]
# Logging level: TRACE (5), DEBUG (10), INFO (20), WARNING (30), ERROR (40), CRITICAL (50)
# TRACE: Very verbose - includes idle checks, session tracking
# DEBUG: Detailed diagnostic information
# INFO: General informational messages
# WARNING: Warning messages
# ERROR: Error messages
# CRITICAL: Critical errors
log_level = INFO
log_file = logs/server.log

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

[VAULT]
# HashiCorp Vault integration for CA key isolation
# Set enabled=false to use traditional file-based keys (current behavior)
# Set enabled=true to use Vault PKI engine (enhanced security with key isolation)
enabled = false

# Vault server connection settings (required when enabled=true)
# For same server: http://127.0.0.1:8200 (dev mode without TLS)
# For separate server: https://192.168.1.20:8200
address = http://127.0.0.1:8200

# Authentication credentials (read from environment variables for security)
# Set these as environment variables: VAULT_ROLE_ID and VAULT_SECRET_ID
role_id = ${VAULT_ROLE_ID}
secret_id = ${VAULT_SECRET_ID}

# PKI mount paths in Vault
pki_rsa_path = pki-subca-rsa
pki_ec_path = pki-subca-ec

# Transit engine path for custom signing operations (OCSP, PQC)
transit_path = transit

# Connection settings
timeout = 30
```



```
retry_attempts = 3

# TLS settings
verify_ssl = true
ca_cert_path =

# Vault roles for different operations
role_scep = scep-enrollment
role_est = est-enrollment
role_default = server-cert

[LDAP]
LDAP_HOST = 172.18.178.24
LDAP_PORT = 389
BASE_DN = dc=rnd-rad,dc=com
PEOPLE_DN = ou=People,dc=rnd-rad,dc=com
ADMIN_DN = cn=Manager,dc=rnd-rad,dc=com
ADMIN_PASSWORD = marketing
enabled = false

[SCEP]
# enable or disable SCEP entirely
enabled = true
# optional path to a file where we persist the serial across restarts
serial_file = pki-subca/serial.txt
# Dump directory for raw SCEP requests (optional)
dump_dir = pki-subca/dumps

# Challenge-password feature
challenge_password_enabled = true
# Validity period for challenge-passwords (default: 60m)
challenge_password_validity = 2m

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

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

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



2.5 Run Server

Shell Command

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

OS Service

Create the main service: pikachu-ca.service

```
sudo tee /etc/systemd/system/pikachu-ca.service > /dev/null << 'EOF'
[Unit]
Description=Pikachu CA Python App
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
User=rocky
Group=rocky
WorkingDirectory=/home/rocky/pki-server-2
ExecStart=/usr/bin/python3.11 /home/rocky/pki-server-2/app.py
Restart=always
RestartSec=5s
Environment=PYTHONUNBUFFERED=1

# Allow binding to ports <1024
AmbientCapabilities=CAP_NET_BIND_SERVICE
CapabilityBoundingSet=CAP_NET_BIND_SERVICE
NoNewPrivileges=true

StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
EOF
```

Create the healthcheck service: pikachu-ca-healthcheck.service

```
sudo tee /etc/systemd/system/pikachu-ca-healthcheck.service > /dev/null <<
'EOF'
[Unit]
Description=Healthcheck for pikachu-ca (restart if https://localhost fails)
After=pikachu-ca.service

[Service]
Type=oneshot
ExecStart=/usr/bin/bash -c '/usr/bin/curl -ksf --max-time 5
https://localhost/ || /usr/bin/systemctl restart pikachu-ca.service'
EOF
```

Create the healthcheck timer: pikachu-ca-healthcheck.timer

```
sudo tee /etc/systemd/system/pikachu-ca-healthcheck.timer > /dev/null <<
'EOF'
[Unit]
Description=Periodic healthcheck for pikachu-ca

[Timer]
OnBootSec=30s
OnUnitActiveSec=60s
Unit=pikachu-ca-healthcheck.service

[Install]
WantedBy=timers.target
EOF
```

Reload systemd to pick up the new units

```
sudo systemctl daemon-reload
```

Enable services & timer at boot



```
sudo systemctl enable pikachu-ca.service
sudo systemctl enable pikachu-ca-healthcheck.timer
```

Start everything now

```
sudo systemctl start pikachu-ca.service
sudo systemctl start pikachu-ca-healthcheck.timer
```

Check status / logs

```
# Main service status
sudo systemctl status pikachu-ca.service

# Timer and healthcheck
sudo systemctl status pikachu-ca-healthcheck.timer
sudo systemctl status pikachu-ca-healthcheck.service

# See timers
systemctl list-timers | grep pikachu

# Logs
journalctl -u pikachu-ca.service -f
journalctl -u pikachu-ca-healthcheck.service -f
```

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

2.6 Ports

The application as development tool can be used in the following ways

Capability	Configuration	Support Technology	Description
------------	---------------	--------------------	-------------

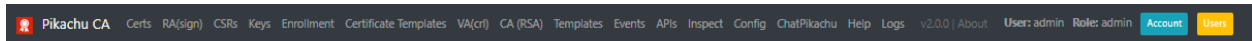


HTTPS without client certificate	[HTTPS] Port = 443	All	One-way TLS Server-authenticated TLS Standard HTTPS
HTTPS that <i>requires</i> client certificates	[TRUSTED_HTTPS] trusted_port = 4443	All	Mutual TLS (mTLS) Two-way TLS TLS with client authentication
HTTP (OCSP)	[DEFAULT] http_port = 80	OCSP	For older technologies
HTTP (SCEP)	[SCEP] http_port = 8090		

3 Features

3.1 Top Navigation Layout

The top navigation bar provides quick access to all major Pikachu CA server functions. Available menu items depend on user role (Admin/User) and authentication mode.



Certs

View, download, revoke, or delete issued certificates.

RA (sign)

Manually sign CSRs to issue certificates.

CSRs

Create, view, download, or delete certificate signing requests.

Keys

Generate, view, download, and delete private keys.

Enrollment

Manage EST/SCEP enrollment policies and extension settings.

Certificate Templates

Create, edit, view, or delete certificate profile templates.

VA (crl)

View revoked certificates and download the CRL.

CA (RSA/EC)

Display root and subordinate CA certificates and chain.

Templates (Admin)



Render new certificate templates from Jinja templates.

Events

Audit log of all system actions (create, delete, revoke, update).

APIs

List all CA APIs including EST, SCEP, Manual RA, CRL, and OCSP.

Inspect

Analyze PEM or Base64 DER data (certs, keys, CSRs, CRLs, etc.).

Config

View the server's active configuration (config.ini).

ChatPikachu

Built-in AI assistant for CA operations.

Help

Open the system help PDF.

Logs

Live server log viewer with filtering.

Version

Displays current server version. v2.0.0

User / Role

Shows logged-in user and role.

Account

Change password or log out.

Users (*Admin*)

Manage user accounts, roles, states, and events.



3.2 Keys Management

List of Keys

The **List of Keys** page displays all keys that have been created in the system and allows users to view, download, or delete them.

<https://pikachu-ca.<sub domain>.com:4443/keys>

List of Keys

[Generate New Key](#)Show: Showing 1-10 of 14[← Previous](#) [Next →](#)

ID	Name	Type ↕	Size/Curve/Algorithm	Created At ↕	User ↕	Actions
17	test18	RSA	2048 bits	2025-12-10 18:12	admin	View Download Delete
16	test	RSA	2048 bits	2025-12-10 18:10	admin	View Download Delete
13	Uzi	RSA	2048 bits	2025-12-10 18:01	pikauser2	View Download Delete
11	test	RSA	2048 bits	2025-12-10 17:59	pikauser2	View Download Delete
10	pikauser2 2	RSA	2048 bits	2025-12-10 17:58	pikauser2	View Download Delete
9	pikauser2	RSA	2048 bits	2025-12-10 17:54	pikauser2	View Download Delete

Page elements

- **Generate New Key** – Button in the top-right corner used to create a new key.
- **Show** dropdown – Controls how many rows are shown per page (for example, 10, 25, 50).
- **Pagination** – *Previous* / *Next* buttons to move between pages when there are more keys than can be shown on a single page.
- **Search / Filter box** – Free-text filter labeled “**Filter by Name, Type, Size, or Date**”. Typing in this field narrows the table to rows matching the entered value.

Table columns

- **ID** – Internal numeric identifier of the key.
- **Name** – Friendly name given to the key (for example, test18, Uzi).
- **Type** – Key type, such as RSA, EC, or PQC (Post-Quantum).



- **Size/Curve/Algorithm** – Indicates key length or algorithm details, for example 2048 bits.
- **Created At** – Date and time when the key was created.
- **User** – Username that owns or created the key.
- **Actions** – Available operations for each key:
 - **View** – Opens a details page for the selected key, where the public and private key material can be inspected (subject to permissions).
 - **Download** – Downloads the key file for backup or use on another system.
 - **Delete** – Permanently removes the key from the system.

Users can sort the table by clicking on sortable column headers such as **Type**, **Created At**, or **User**, making it easy to locate specific keys or review recently created ones.

View Key

The **View Key** page displays detailed information about a specific key selected from the **List of Keys** table. This page allows users to inspect both the public and private portions of the key for reference or export.



Key Details (ID: 2)

Name: HTTPS-CLEINT

Type: EC

Curve: prime256v1

Created At: 2025-06-18 14:55

Public Key

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEt2wW0g+tpyqucfxcurs6jNm1vAJ9
kCS1xw7oGOZZzicj9toOVo9oJ7Hin+fpRGKJRxKkFNWYrGw2TO1oW3Wssw==
-----END PUBLIC KEY-----
```

Private Key

```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEII2mYCFca40QGzPGU2HypfS83jJs/szpSQqvmkmtDysGoAoGCCqGSM49
AwEHoUQDQgAEt2wW0g+tpyqucfxcurs6jNm1vAJ9kCS1xw7oGOZZzicj9toOVo9o
J7Hin+fpRGKJRxKkFNWYrGw2TO1oW3Wssw==
-----END EC PRIVATE KEY-----
```

[Back to Keys List](#)

Key Details Displayed

- **Key ID**
The internal identifier of the key (e.g., *ID: 2*).
- **Name**
The user-assigned label for the key (e.g., *HTTPS-CLIENT*).
- **Type**
Indicates whether the key is **RSA**, **EC**, or **PQC**.
- **Curve / Size / Algorithm**
Displays key parameters depending on type:
 - EC keys show the selected curve (e.g., *prime256v1*).
 - RSA keys show the key size (e.g., *2048 bits*).



- PQC keys show the algorithm (e.g., *mlds44*).
- **Created At**
The date and time when the key was generated.

Public Key

A read-only text block showing the full PEM-encoded public key.

This key can be copied and used in CSRs, device configuration, or other cryptographic operations.

-----BEGIN PUBLIC KEY-----

...

-----END PUBLIC KEY-----

Private Key

A read-only text block showing the PEM-encoded private key.

Users can copy the key if needed for externally managed signing operations, devices, or testing tools.

-----BEGIN EC PRIVATE KEY-----

...

-----END EC PRIVATE KEY-----

Security Note:

Private keys displayed here should be handled carefully.

Exporting or copying private keys should be restricted to authorized users only.

Generate a New Key

The Generate a New Key page allows users to create a new cryptographic key that can later be used for generating CSRs and issuing certificates.

<https://pikachu-ca.<sub domain>.com:4443/generate>



Generate a New Key

Key Name:

KeyGen2

Key Type:

EC

Curve (for EC):

prime256v1 (secp256r1)

Generate Key

Creation Fields on the page:

Key Name

A user-defined label for the new key.

This name will appear in the List of Keys table and helps identify the key for future use (e.g., KeyGen2).

Key Type

A dropdown menu used to select the type of key to generate.

Available options typically include:

1. RSA Keys

Widely supported and commonly used for TLS and device authentication.

Supported key sizes:

- a. 2048 bits – Standard security level
- b. 4096 bits – Higher security for long-term or sensitive deployments

2. EC (Elliptic Curve) Keys

EC keys provide strong security with smaller key sizes and better performance.

When EC is selected, the Curve dropdown appears, offering the following options:

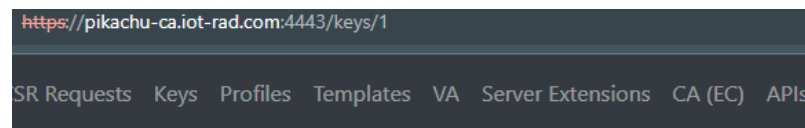
- a. prime256v1 (secp256r1) – Most widely used NIST P-256 curve
- b. secp384r1 – Higher-security NIST P-384 curve
- c. secp521r1 – Highest NIST-level EC security
- d. secp256k1 – K-256 curve, commonly used in blockchain applications

3. PQC (Post-Quantum Cryptography) Keys

Post-quantum algorithms designed to resist attacks from future quantum computers. When PQC is selected, the PQC Algorithm dropdown appears with:

- mlds44 (Dilithium2 / NIST L1) – Baseline PQ security
- mlds65 (Dilithium3 / NIST L3) – Stronger PQ protection
- mlds87 (Dilithium5 / NIST L5) – Highest standard PQ strength

These algorithms require an environment configured with OpenSSL 3.x + oqsprovider.



Key Details (ID: 1)

Name: key

Type: EC

Curve: prime256v1

Created At: 2025-04-15 16:37

Public Key

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEb7YxnbXgqkNfWpORxCWr2RXEVN3z
tc/wL2f5H9fD130C4XMba1V070Mna9MxNIvuX5jLOiL/hwR0Z5jR7uQk+w==
-----END PUBLIC KEY-----
```

Private Key

```
-----BEGIN EC PRIVATE KEY-----
MHCCAQEEIFgMuETAaTdX2UdWBKxdS4bXAsFOVaQBe3hNPSPmQ8ToAoGCCqGSM49
AwEHoUQDQgAEb7YxnbXgqkNfWpORxCWr2RXEVN3ztc/wL2f5H9fD130C4XMba1V0
70Mna9MxNIvuX5jLOiL/hwR0Z5jR7uQk+w==
-----END EC PRIVATE KEY-----
```

[Back to Keys List](#)



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

3.3 Certificate Templates

Certificate Templates list

The **Certificate Templates** page displays all certificate templates available on the server. Templates define the structure and attributes used when generating certificate profiles, including X.509 extensions and Distinguished Name (DN) fields. Templates may be created manually or derived from predefined Jinja-based template files.

Certificate Templates

Generate New Certificate Template

Show: 10 Showing 1-10 of 15

← Previous Next →

Filter by Template Name, Source, Type, or Date

ext X509 Extensions | **req** Certificate DN (Distinguished Name)

Name	Originating Template	Profile Type %	Created At %	User %	Usage	Actions
https-test	manually	HTTPS	2025-12-09 15:10	admin	req	View Edit Delete
event_template	manually	UZI_TESTS	2025-12-08 20:46	admin	req	View Edit Delete
kuku.cnf	manually	HTTPS		N/A	req	View Edit Delete
mocana_client_ext.cnf	mocana_client_ext.cnf.j2	Mocana		N/A	ext	View Edit Delete
mocana_client_req.cnf	mocana_client_req.cnf.j2	Mocana		N/A	req	View Edit Delete
mocana_client_request.cnf	mocana_client_request.cnf.j2	Mocana		N/A	req	View Edit Delete
bitbucket_server.cnf	6w_ca_root.cnf.j2	HTTPS		N/A	req	View Edit Delete
bitbucket_server_ext.cnf	ssl_server_ext.cnf.j2	SERVER_EXT		N/A	ext	View Edit Delete
ssl_server_ext.cnf	ssl_server_ext.cnf.j2	HTTPS		N/A	ext	View Edit Delete

- **Show** — Adjusts the number of displayed rows (e.g., 10, 25, 50).
- **Pagination Controls** — *Previous* / *Next* buttons navigate through multiple pages.
- **Filter Box** — Searches templates by name, source, type, or date.

At the top of the page **Generate New Certificate Template** Creates a new template manually or based on an existing template file.

Below the toolbar, the page highlights two major categories of template content:

- **X.509 Extensions**
- **Certificate DN (Distinguished Name)**

These labels indicate the two primary components used when rendering certificate profiles.

Template List Columns

Each row in the table represents an existing certificate template. The columns are:

- **Name**
The template's identifier (e.g., *https-test*, *event_template*, *mocana_client_req.cnf*).
- **Originating Template**
Indicates the source, such as:
 - **manually** — Created by a user from scratch
 - **filename.j2** — Generated from a Jinja certificate template file
- **Profile Type**
Specifies the intended usage or category of the template (e.g., *HTTPS*, *Mocana*, *UZI_TESTS*).
- **Created At**
Date and time when the template was created.
- **User**
The user who created the template (e.g., *admin*, *N/A* for system-generated templates).
- **Usage**
Indicates which parts the template provides:
 - **req** — Provides CSR configuration
 - **ext** — Provides X.509 extension configuration
(Some templates provide both depending on their design.)
- **Actions**
Available operations for managing templates:
 - **View** — Opens a detailed read-only view of the template content

- **Edit** — Opens the template in editor mode for modification
- **Delete** — Removes the template from the system

Typical Use Cases

- Creating customized CSR templates for device onboarding
- Managing extension templates for specific certificate enrolment methods
- Revising DN structures for organizational standards
- Maintaining reusable templates that simplify certificate profile creation

Generate New Certificate Template

The **Generate New Certificate Template** page allows users to create a new certificate template that can later be used to generate profiles for CSRs or X.509 extensions. Templates may be created manually or derived from existing template files, depending on the workflow.

Create Profile

Profile Name:

https_client

Profile Type:

HTTPS

Profile Content:

```
[ v3_ext ]
basicConstraints = critical, CA:FALSE
keyUsage = critical, digitalSignature
extendedKeyUsage = clientAuth
authorityInfoAccess = @aia_section
crlDistributionPoints = @crl_section

[ aia_section ]
OCSP;URI.0 = http://pikachu-ca.safe-room/ocsp
caIssuers;URI.0 = https://bitbucket.safe-room/ocsp

[ crl_section ]
URI.0 = https://pikachu-ca.safe-room/downloads/crl
```

Create

Fields on the Page

1. Profile Name

A user-defined label for the new template.

This name will appear in the **Certificate Templates** list for easy identification.

2. Profile Type

Indicates the intended use of the template (e.g., *HTTPS, Mocana, Client*, etc.).

This classification helps organize templates and associate them with specific certificate workflows.

3. Profile Content

A free-form text area where the user enters the configuration for the certificate template.

The system determines the type of template based on the sections included in the content:

- CSR Configuration Templates ([req])

If the content contains a **[req]** section, the system identifies the template as a **CSR configuration template**.

Such templates define:

- Distinguished Name (DN) fields
- Request attributes
- Key parameters
- Required input for certificate signing

Use this format when creating templates intended for CSR generation.

- 2. X.509 Extension Templates ([v3_ext])

If the content contains a **[v3_ext]** section, the system identifies it as an **X.509 extension configuration template**.

These templates specify certificate extensions such as:

- Key usage
- Extended key usage
- Authority Info Access (AIA)
- Subject Alternative Names (SAN)
- CRL and OCSP endpoints



Use this format for templates that provide certificate extensions during certificate issuance.

View Certificate Template

Certificate Template File: https-test

Originating Template:

Template Type: HTTPS

```
[ req ]
prompt          = no
distinguished_name = dn
default_md       = sha256
default_bits     = 2048
default_ec_curve  = prime256v1

[ dn ]
C = IL
ST = IL
L = TLV
O = RAD
OU = RND
CN = https-est-secure-test
```

[Back to Certificate Templates](#)

The **View Certificate Template** page shows the full configuration and basic details of a selected template.

- **Template Name** and **Type** are displayed at the top, along with the **Originating Template** (manual or based on a .j2 file).
- The **Template Content** appears in a read-only text block.
 - A template containing **[req]** defines CSR settings.
 - A template containing **[v3_ext]** defines X.509 extension settings.



Use **Back to Certificate Templates** to return to the main list.

Edit Certificate Template

The **Edit Profile** page allows users to modify the content of an existing certificate profile.

- The **Profile Content** field displays the current configuration in an editable text area.
- Users may update CSR-related sections ([req]) or X.509 extension sections ([v3_ext]) depending on the profile's purpose.
- After making changes, click **Save** to update the profile and return to the profile list.

3.4 Enrollment

The **Enrollment** page displays all defined enrollment policies.

These policies control certificate validity, required extensions, and how CSR enrollment is handled for various use cases such as HTTPS, EST, SCEP, and system-level policies.



Enrollment Policies List

Enrollment Policies

Policies include extension config and validity; future fields like restrictions will appear here.

Challenge Passwords

Generate New Enrollment Policy

Show: 10 Showing 1-4 of 4

← Previous

Next →

Filter by any column

Name ↕	Uses ↕	User ↕	Validity (days) ↕	Created (Updated) ↕	Extension ↕	Actions
HTTPS SERVER WIN TEST		admin	365	2025-12-09 15:07:34 (2025-12-09 15:29:41)	[v3_ext] subjectAltName = @alt_names [alt_names] DNS.1 = localhost DNS.2 = localhost-wsl-win IP.1 = 127.0.0.1	Edit Delete
event_policy	EST	admin	100	2025-12-08 20:44:57 (2025-12-08 20:45:17)	[v3_ext] basicConstraints = critical, CA:FALSE subjectKeyIdentifier = hash authorityKeyIdentifier = keyid,issuer authorityInfoAccess = @aia_section crlDistributionPoints = @c...	Edit Delete
admin	SCEP	admin	10	2025-12-07 11:40:51 (2025-12-07 11:41:42)	[v3_ext] basicConstraints = critical, CA:FALSE subjectKeyIdentifier = hash authorityKeyIdentifier = keyid,issuer	Edit Delete
server_ext.cnf	System	-	365	2025-12-07 11:39:49	[v3_ext] basicConstraints = critical, CA:FALSE subjectKeyIdentifier = hash authorityKeyIdentifier = keyid,issuer	Edit Delete

Page Features

- **Show** — Select how many policies to display per page.
- **Pagination** — *Previous* / *Next* buttons navigate through multiple pages.
- **Filter Box** — Allows filtering by any column (name, user, extension type, etc.).
- **Challenge Passwords** — Opens configuration for managing CSR challenge passwords.
- **Generate New Enrollment Policy** — Creates a new policy with custom validity and extension rules.

Table Columns

Each row represents an enrollment policy and includes:

- **Name**
Human-readable policy name (e.g., *HTTPS SERVER WIN TEST*, *event_policy*).
- **Uses**
Indicates the enrollment flow the policy applies to, such as:



- **EST** – Used for EST-based enrollments
 - **SCEP** – Used for SCEP requests
 - **System** – Built-in default or server-level policy
- **User**
The user who created or owns the policy (e.g., *admin*).
- **Validity (days)**
The certificate validity period enforced by this policy.
- **Created (Updated)**
Shows when the policy was created and, if applicable, last modified.
- **Extension**
Displays the X.509 extension block associated with the policy, such as:
 - [v3_ext] definitions
 - Adding Subject Alternative Name (SAN)
 - Adding Authority Information Access (AIA)
 - Adding Key Usage / Extended Key Usage (KU / EKU)
 - Adding Authority Information Access (AIA) for OCSP
 - Adding CRL Distribution Point (CDP) URL
 - Adding CA Issuers URL for retrieving the CA certificate chain

These fields define what extensions will be added to certificates issued under this policy.

- **Actions**
 - **Edit** — Modify the policy's validity, usage, or X.509 extension content.
 - **Delete** — Remove the policy from the system.

Generate Enrollment Policy

The **New Enrollment Policy** page allows users to create a policy that defines how certificates will be issued for a specific enrollment workflow (EST, SCEP, or system-wide usage).



New Enrollment Policy

Enrollment Policy Name

est_deviceX

Validity (days)

365

☐ System enrollment policy (visible to all)

☒ Use for EST

☐ Use for SCEP

Load extension from certificate template (optional)

mocana_client_ext.cnf



Load into form

Extension configuration

```
[ v3_ext ]
basicConstraints = critical, CA:FALSE
keyUsage       = critical, digitalSignature
extendedKeyUsage = clientAuth
authorityInfoAccess = @aia_section
crlDistributionPoints = @crl_section

[ aia_section ]
```

Create Policy

Cancel

Fields on the Page

- Enrollment Policy Name

A user-chosen name that identifies the policy (e.g., est_deviceX).
This name appears in the Enrollment Policies list.

- Validity (days)

Defines how long the certificates issued under this policy will remain valid.

- System Enrollment Policy

When enabled, the policy becomes visible and usable by all users.

- Use for EST / Use for SCEP

Specifies which enrollment method(s) the policy applies to:

- Use for EST — Policy applies to EST enrollment requests.
- Use for SCEP — Policy applies to SCEP enrollment requests.
(One or both options can be selected depending on requirements.)



- Load Extension From Certificate Template (optional)

Users may optionally select an existing certificate template.

Click Load into form to insert its extension block into the policy editor automatically.

This helps quickly reuse predefined X.509 extension settings.

- Extension Configuration

A text area containing the X.509 extension definitions to apply when signing certificates.

Typical sections include:

- [v3_ext] — Defines X.509 extensions such as key usage, extended key usage, AIA, CRL distribution points, etc.

Additional sections (e.g., [aia_section], [crl_section]) may appear depending on the template.

Users may edit these settings manually before creating the policy.

Create / Cancel

- Create Policy — Saves the new enrollment policy and returns to the policy list.
- Cancel — Discards changes and returns without creating a policy.

Edit Enrollment Policy

The **Edit Enrollment Policy** page allows users to update an existing enrollment policy.

Users can modify:

- **Policy Name**
- **Validity (days)**
- Whether the policy is **system-wide**, or used for **EST** or **SCEP**
- **Extension configuration**, including [v3_ext] and any supporting sections

An optional certificate template may be selected and loaded to replace or update the extension block.

Click **Save Changes** to apply updates or **Cancel** to return without saving.

Challenge Passwords

The **Challenge Passwords** page lists all challenge-password tokens generated for certificate enrollment. These one-time passwords are typically used in SCEP or CSR-based workflows to authenticate enrollment requests.

Challenge Passwords

[Generate New Challenge-Password](#)

Show: Showing 1-10 of 14

[← Previous](#) [Next →](#)

Filter by Password, User, or Date

[Delete All My Expired](#)

[Delete All Expired \(System\)](#)

Challenge-Password <small>(click to copy)</small> ↑	Status ↑	User ↑	Created (local) ↑	Validity ↑	Expires (local) ↑	Actions ↑
B2D18DD0A468BF3D8755A63D5034DE86	Available	admin	2025-12-12 15:18	2m	2025-12-12 15:20	
4EASF1D736519E57E0E4B1DEB3EA3CE6	Available	admin	2025-12-12 15:18	2m	2025-12-12 15:20	
0F0B3AB03E7E42AE6EB333D74C06D620	Consumed	admin	2025-12-11 10:14	2m	2025-12-11 10:16	
4C0826C81A9B76ABD32E1BA40365A63C	Expired	admin	2025-12-10 16:23	2m	2025-12-10 16:25	Delete
262212D641FC295887B51E76B9378A75	Consumed	admin	2025-12-10 11:18	2m	2025-12-10 11:20	
34D987801496912D107E959151FFAA2E	Consumed	mayagolan	2025-12-09 13:40	2m	2025-12-09 13:42	
34728D79595A1168F89144D52F16C230	Consumed	admin	2025-12-09 13:39	2m	2025-12-09 13:41	
E463C3C50E98EE402AE6798BE84CD139	Consumed	admin	2025-12-08 23:02	2m	2025-12-08 23:04	

Page Features

- **Generate New Challenge-Password**
Creates a new one-time password with a short validity window.
- **Show / Pagination**
Adjusts the number of rows displayed and allows navigation through pages.
- **Filter Bar**
Filters entries by password value, user, or date.
- **Delete All My Expired**
Removes all expired challenge passwords belonging to the current user.



- **Delete All Expired (System)**
Removes all expired passwords for all users (admin only).

Table Columns

- **Challenge-Password** *(click to copy)*
Displays the generated token. Clicking copies it to the clipboard.
- **Status**
Indicates the state of each token:
 - **Available** – Valid and unused
 - **Consumed** – Already used for enrollment
 - **Expired** – No longer usable
- **User**
The user who created the challenge password.
- **Created (local)**
Local server timestamp of when the password was generated.
- **Validity**
Duration for which the password is valid (e.g., 60m).
- **Expires (local)**
Local time when the password becomes invalid.
- **Actions**
 - **Delete** – Removes the selected password (typically used for expired or consumed passwords).

The profile can be created either by editing or rendered from template



3.5 Events (System)

The **Manage Events** page provides a complete audit log of all system-level actions performed on resources such as certificates, keys, profiles, and enrollment policies. It is designed to help administrators track operations, troubleshoot changes, and maintain compliance visibility.

Manage Events

Resource Type

Event Type

Resource Name

Filter

Clear

Show: 10 Showing 1-10 of 102

← Previous

Next →

Filter by Event Type, Resource, Name, or Details

Time	User	Event Type	Resource Type	Resource Name	Details	
2025-12-12 14:02	admin	Delete	Certificate	0xbf71439564b11b18	{"subject": "commonName=estclient-curl-mtls-0:4:10:12:2025", "issued_via": "est", "rowcount": 1}	View
2025-12-12 14:01	admin	Delete	Certificate	0x573a7f1adc32fb12	{"subject": "commonName=estclient-curl-mtls-0:7:10:12:2025", "issued_via": "est", "rowcount": 1}	View
2025-12-12 14:01	admin	Delete	Certificate	0x2e053f2ad6fd708a0f...	{"subject": "commonName=sscep-test-pass-10:15:11:12:2025", "issued_via": "scep", "rowcount": 1}	View
2025-12-12 14:01	admin	Revoke	Certificate	0x2e053f2ad6fd708a0f...	{"subject": "commonName=sscep-test-pass-10:15:11:12:2025"}	View
2025-12-12 13:18	admin	Create	challenge_password	B2D18DD0A46B8F3D8755...	{"validity": "2m"}	View
2025-12-12 13:18	admin	Create	challenge_password	4EA5F1D736519E57E0E4...	{"validity": "2m"}	View
2025-12-11 08:15	admin	Create	Certificate	0x2e053f2ad6fd708a0f...	{"subject": "commonName=sscep-test-pass-10:15:11:12:2025", "issued_via": "scep"}	View

Legend:

Event Types

- Create Create
- Delete Delete
- Revoke Revoke
- Update Update

Resource Types

- Certificate Certificate
- Policy Policy
- Key Key
- Profile Profile
- Challenge Password Challenge Password

Other

- Other Other/Unknown

Filtering

You can filter events using:

- **Resource Type**
(Certificate, Key, Profile, Policy, Challenge Password, Other)
- **Event Type**
(Create, Delete, Revoke, Update)
- **Resource Name**
Filter by subject, serial, name, or identifier.
- **Search Bar**
Free-text search across event type, resource name, and details.

Pagination controls allow browsing through large event histories.

Table Columns

- **Time**
Exact timestamp of the event.
- **User**
The user who performed the action (actor).
- **Event Type**
Categorizes the operation:
 - **Create** – A new resource was generated
 - **Delete** – A resource was removed
 - **Revoke** – A certificate was revoked
 - **Update** – A resource was modified
- **Resource Type**
Indicates the kind of object affected:
 - Certificate
 - Key
 - Profile
 - Policy



- Challenge Password
 - Other / Unknown
- **Resource Name**
Identifier of the resource (e.g., certificate serial, key name, policy name).
- **Details**
Additional JSON metadata, such as:
 - Subject DN
 - Issuance method (est, scep)
 - Validity fields
 - Row counts or system-side info
- **View**
Opens a detailed view of the event's full log data.

Legend

Event Types

- **Create** – Resource created
- **Delete** – Resource deleted
- **Revoke** – Certificate revoked
- **Update** – Resource modified

Resource Types

- **Certificate**
- **Policy**
- **Key**
- **Profile**
- **Challenge Password**
- **Other** – Unclassified/unknown events



3.6 Templates (admin role only)

Loading/Editing

The X509 Certificate Templates page allows users to work with Jinja-based certificate template files (.cnf.j2). These templates contain placeholders for CA information and certificate parameters that must be filled in before generating a usable profile.

X509 Certificate Templates

Choose a template:

Template Actions

- **Choose a template**
Select a .cnf.j2 template from the dropdown menu.
- **Load Template**
After selecting a template, clicking **Load Template** opens the **Fill in Variables** form.
This form displays all variables defined inside the template

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

ca_city:

ca_common_name:

ca_country:

ca_organization:

ca_organizational_unit:

ca_state:

Profile Name:

Profile Type:

Rendering

- Render Profile**
Generates a new certificate template using the filled-in values and the selected Jinja template. The created template appears in the Templates list, where the *Originating Template* column shows which .cnf.j2 file it was generated from.
- Back to Template List**
Returns to the template selection page.

View Rendered Certificate Template

Rendered Certificate Template:

Originating Template: 6w_ca_root.cnf.j2

Template Type: HTTPS

Certificate Template Content

```
# CA Certificate Configuration Template for Root ECC Certificate
[ 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
```

[Edit](#)
[Back to Template List](#)
[View All Certificate Templates](#)

Creating New Template

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
```

3.7 Certificates

List

The **Issued Certificates** page lists all certificates that have been signed and issued by the CA. Users can filter, inspect, download, revoke, or delete certificates depending on their role and permissions.



Issued Certificates

Show: 10 Showing 1-10 of 94

[← Previous](#) [Next →](#)

Filter by Common Name, Serial, Key or Date

ID	Common Name	Serial <small>(click to copy)</small>	Key <small>↑↓</small>	Date (UTC) <small>↑↓</small>	Status <small>↑↓</small>	Enrollment Method	User <small>↑↓</small>	Actions
97	sscep-test-pass-10:15:11:12:2025	0x2e053f2ad6fd708a0f...	RSA/2048	2025-12-11 10:15	Valid	SCEP	admin	View Download PFX Revoke Delete
96	https-est-secure-test	0x965fcb3b4a781339	RSA/2048	2025-12-11 10:11	Revoked	UI	admin	View Download PFX Revoked Delete
95	sscep-test-pass-11:20:10:12:2025	0x2e053f2ad6fd708a0f...	RSA/2048	2025-12-10 11:20	Valid	SCEP	admin	View Download PFX Revoke Delete
94	estclient-curl-mtls-0:7:10:12:2025	0x573a7f1adc32fb12	EC/prime256v1	2025-12-10 00:07	Valid	EST	N/A	View Download PFX Revoke Delete

Page Features

- **Show / Pagination**
Controls how many certificates are displayed and allows navigation between pages.
- **Filter Bar**
Filters certificates by common name, serial number, key type, date, or any visible column.

Table Columns

Each row represents an issued certificate and displays the following information:

- **ID**
Internal numeric identifier for the certificate.
- **Common Name**
The CN value from the certificate's Distinguished Name.
Clicking the name opens the certificate detail view.
- **Serial (click to copy)**
The certificate's serial number.
Can be copied for debugging, OCSP checks, or revocation workflows.
- **Key**
The key algorithm and size used for the certificate:
 - RSAs

- ECs.
 - PQCs (if applicable)
- **Date**
The date and time the certificate was issued.
- **Status**
Indicates whether the certificate is:
 - **Valid**
 - **Revoked**
 - **Expired**
- **Enrollment Method**
Identifies how the certificate was issued:
 - **SCEP** – Certificate issued via SCEP enrollment
 - **UI** – Manually signed via the web UI
 - **EST** – Certificate issued via EST enrollment
- **User**
The user or system component that performed the signing. (admin user only)

Actions

- **View**
Opens full certificate details (subject, issuer, extensions, validity period, etc.).
- **Download**
Downloads the certificate chain:
 - End-entity certificate
 - Intermediate certificate
 - Root certificate
- **PFX (*secret-protected*)**
Generates a PKCS#12 bundle containing:
 - Certificate
 - Private key
 - Intermediate CA

Suitable for importing into Windows, browsers, or secure devices.



Applicable only if the Certificate original Key is available

Action execution popup dialog for the needed secret

- **Revoke**
Invalidates the certificate and adds it to the CRL. Requires typing “**revoke**” to confirm.
- **Delete**
Removes the certificate from the database. Requires typing “**delete**” to confirm.

View

The **View Certificate** page displays the full details of an issued certificate, including a human-readable summary, the full parsed certificate text, and the raw PEM-encoded certificate. This view is typically accessed from the *Issued Certificates* table by selecting the **View** action.

Certificate Details

Issued via: SCEP

Certificate Summary

Public Key Algorithm: RSA
Public Key Parameters: 2048 bits
Subject: commonName: sscep-test-pass-10:15:11:12:2025
Issuer: countryName: IL stateOrProvinceName: TLV localityName: Tel Aviv organizationName: RAD organizationalUnitName: RD commonName: RADSubTestRSA
Serial Number: 0x2e053f2ad6fd708a0f4792cb7de14b7a89f3bcaa
Version: v3
Not Valid Before: 2025-12-11 08:15Z
Not Valid After: 2025-12-21 08:15Z
Signature Algorithm: sha256WithRSAEncryption

Detailed Certificate Text

```
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number:
    2e:05:3f:2a:d6:fd:70:8a:0f:47:92:cb:7d:e1:4b:7a:89:f3:bc
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C=IL, ST=TLV, L=Tel Aviv, O=RAD, OU=RD, CN=RADSubTestRSA
  Validity
    Not Before: Dec 11 08:15:37 2025 GMT
    Not After : Dec 21 08:15:37 2025 GMT
  Subject: CN=sscep-test-pass-10:15:11:12:2025
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (2048 bit)
    Modulus:
      00:9e:5c:24:86:f4:eb:f0:f2:0b:5f:56:d4:a1:5a:
      5b:fc:fe:eb:d2:13:9e:2a:1e:0b:a6:f0:20:b2:3a:
      42:a7:46:90:f6:29:9b:ae:eb:0e:64:a9:a4:78:ec:
      bf:6c:0e:01:0c:19:19:7f:33:b7:39:0c:a1:a9:68:
      7b:7f:58:f7:f3:16:d0:cb:73:d5:34:82:b6:08:99:
      b2:ae:fb:18:8c:94:32:0d:b7:2c:83:a0:48:fa:17:
      94:07:c3:ae:91:53:50:d5:be:05:88:3d:4c:55:ad:
      70:65:fc:56:90:ee:5d:77:34:dc:b1:72:5d:22:c2:
      42:49:f5:46:f6:59:7e:40:ec:d5:c6:9e:e0:71:e3:
      97:e8:4d:0a:95:2b:07:db:63:73:9f:92:37:0e:19:
      b4:0d:61:68:71:6a:ab:24:1a:82:2e:0c:9c:20:1d:
      7e:c1:3f:7b:46:2d:b2:b1:c3:7a:06:fb:80:0c:85:
      cc:87:a3:f4:b9:ab:ff:4a:0b:f5:40:cf:fc:b6:ec:
      a7:b4:e7:aa:f9:1a:b2:af:96:fa:5a:85:64:82:27:
```



Raw Certificate (PEM Format)

```
-----BEGIN CERTIFICATE-----
MIIEaJCCA1KgAwIBAgIUlgU/Ktb9cIoPR5LLfELeonzvKkwDQYJKoZIhvcNAQEL
BQAwYTELMakGA1UEBhMCSUwxDQAKBgNVBAGMA1RMVjERMA8GA1UEBwwIVGVsIEF2
aXVxODAKBgNVBAoMA1JBRDELMAkGA1UECwwCUkxjFjAUBGNVBAHMDVJBRFN1Y1R1
c3R5SU0EWWhcNMjUxMjEwMDkyMDEzWWhcNMjUxMjEwMDkyMDEzWjArMSkwJwYDVQ
Q0DCB2c2N1c10ZXN0LXBhc3MtMTE6MTA6MTI6MTA6NTCCAS1wDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAAK0a53wub4hHG7v7vRDcdaPDLZUNVH2Jnz8QsPt
wTecJ0wJnQj/uLIxzvOC12HNEHF104qKLYS4uuR6Aaxa103q0YdsInRCVINGofS1
n3uQLge1n7+xBaL3mP1S9vfrxJ3yV914++vPxs1EINqCZwRdM1e76CY5q3b33gVO
N6qZx3WKI+qPvgJ9u1QNm+w0112a/1t01Rs/9FLnWYzS87FuaN5dc3TXnnvU5ydQ
jkrUva1abmRqAeZGZUmwK1dUv0bNkpPG7p/38059IKNq11T+tps5WfUaAnZ0DEp
0t8ka0VR1fzuJcZKwawsV5A6kG6Pktntg/PBV91umRq1qxcCAwEAAANQME4wDAYD
VR0TAQH/BAIwADAdBgNVHQ4EFgQUVb+ULg8g5rT+12rWx+nI7BmAsrUwHwYDVR0j
BBgwFoAUGqAk+aJsta0ewXGhKzDsIxkykEQwDQYJKoZIhvcNAQELBQADggIBAHRj
rRUTzycj1Hc/4/Fn597TTL9tIR++zqHwJ3T/Wtx86f5W47fxZLdbIuaJwSwUZZzC
BnEIS10NnVk2Bg1AcvR7VvZUJx07Cpu3rZJe1RjC9qL28DopqSeUu8ZfdJZyRvNt
E/Ehk3bv88iQRQCV1Q00n/e29VMorPIBCx0h3GL5+szw200w63m+d4/4WgCcegI6
n5fSEKvWpHj/HKSK42yrXssqGDYNMh1m6NYOWWYTDvMHJUL6tT8DC64ex1ZU4JPM
pdL2T1YFqsXXCaNYkMRSDN3szh+zURzuA5cFy9B+SYZpnQ1cJ+t1vRF3mWLRNqo
0/VPzbnrLEqfrtt34D43U0tCjoNcnbXkR3FZNImuJbIE6/JX6bnjXUzXZK10Txg
fLBQ0mOwsT+HIApFvN5I45346Du1qVWuaCKUjhcxtg/tLaCvQv2fsg1FjyvVEvp
ztw7gRqdLjdIV20jbepQ1pDs3SnoKVCQRNwx30hDbjr1U/Nb7qcokD3LKcZwJZrR
q7pTG1RyonyS3T1kGf/3Pz4X83gwWkdRBxN7/zko0x1xpDNuMGBMn2NZUATA+1bu
x1LYwcd5f75NokZ9WqX5G26K/R6fMV0RyxybCk+T/XvpwFwz2wkZQFmX61WHBA6H
Imo/dANzSMIPyY9Ko/yDI7yapn3CYS46dPN5jCud
-----END CERTIFICATE-----
```

[← Back to Home](#)

Certificate Summary

The left-hand panel provides a structured summary of the certificate's main attributes:

- **Public Key Algorithm**
Displays the algorithm used (e.g., RSA, EC, PQC).
- **Public Key Parameters**
Shows key size or curve (e.g., RSA/2048 bits, EC/prime256v1).
- **Subject**
Contains the subject DN fields such as commonName.
- **Issuer**
Displays the issuing CA's distinguished name.
- **Serial Number**
Full serial number in hexadecimal format.
- **Version**
Certificate X.509 version (typically v3).
- **Validity**
 - **Not Valid Before**
 - **Not Valid After**



- **Signature Algorithm**
Displays the signature algorithm used by the issuing CA.

At the top of the page, a badge indicates the **Enrollment Method**, such as:
Issued via: SCEP, UI, or EST.

Detailed Certificate Text

The right-hand panel shows the full certificate as parsed by OpenSSL. This includes:

- Certificate version and serial number
- Signature algorithm
- Issuer and subject data
- Validity dates
- Public key details (algorithm, bit size, modulus, etc.)
- Full X.509 extensions, if present

This view is useful for administrators validating certificate structure or debugging enrollment issues.

Raw Certificate (PEM Format)

The bottom section displays the certificate in standard PEM format:

```
-----BEGIN CERTIFICATE-----
```

```
MIIB...
```

```
...
```

```
-----END CERTIFICATE-----
```

This representation can be copied for:

- Import into external systems
- Manual verification
- OCSP checks
- Certificate decoding tools

Navigation

- **Back to Home** returns the user to the main dashboard or previous certificate list.

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

1. View – Display the certificate's details and metadata.
2. Download – Retrieve the certificate file for local use or storage.
3. PFX – Generate and download a PFX/PKCS#12 bundle containing the certificate and private key.
4. Revoke – Mark the certificate as invalid so it can no longer be trusted or used.
5. Delete – Permanently remove the certificate record from the system.

The Actions Delete, revoke and PFX are secret protected

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

3.8 CSRs

List

The **CSR List** page displays all certificate signing requests submitted through the system. Users can view, download, or delete pending CSRs before they are signed into certificates.



List of Certificate signing request (CSR)

[Generate New CSR](#)

Show: 10 Showing 1-10 of 14

[← Previous](#) [Next →](#)

Filter by Name, Key, Profile, or Date

ID	Name	Key	Profile	Created At %	User %	Actions
16	yoni	test18		2025-12-11 08:10	admin	Download View Delete
15	https-test	HTTPS-CERT		2025-12-09 15:12	admin	Download View Delete
14	event_csr	test		2025-12-08 20:38	admin	Download View Delete
13	YOSSI	HTTPS-CLEINT		2025-11-23 12:21	N/A	Download View Delete
10	mocana (RSA) 3	mocana		2025-11-17 18:08	N/A	Download View Delete
9	mocana (RSA) 2	mocana		2025-11-17 13:53	N/A	Download View Delete
8	mocana (RSA)	mocana		2025-11-13 14:38	N/A	Download View Delete

Page Features

- **Generate New CSR**
Opens the form to create a new certificate signing request using an existing key and profile.
- **Show / Pagination**
Controls how many CSRs are displayed and allows navigation through pages.
- **Filter Bar**
Filters CSRs by name, key, profile, user, or date.

Table Columns

- **ID**
Internal identifier for the request.
- **Name**
The user-assigned name for the CSR.
- **Key**
The key used to generate the CSR.
- **Profile**
The certificate profile applied during CSR creation.
- **Created At**
Timestamp of when the CSR was generated.



- **User**
The user who created the CSR. (Visible only to users with the admin role.)

Actions

- **Download**
Downloads the CSR in PEM format.
- **View**
Displays the full CSR details, including subject information and public key.
- **Delete**
Removes the CSR from the system.

View

The **View CSR** page displays the details of a selected certificate signing request.



CSR Details (ID: 15)

Name: https-test

Key: HTTPS-CERT

Profile:

Profile Details

Template Name:

Profile Configuration Content

```
[ req ]
prompt          = no
distinguished_name = dn
default_md       = sha256
default_bits     = 2048
default_ec_curve  = prime256v1

[ dn ]
C = IL
ST = IL
L = TLV
O = RAD
OU = RND
CN = https-est-secure-test
```

Created At: 2025-12-09 15:12

CSR (PEM)

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBHzCBKgIBADBlMQswCQYDVQQGEwJ3TDELMARGA1UECAwCSUxxDDAKBgNVBAMM
A1R9VjEENMAoGA1UECgwDUkFEHQswCgYDVQQLDANSUkFEHQswCgYDVQQLDANSUkFE
LWVzZC11Z2VhbnR1cmUtZGVzdB8ZMBGByqGSM49AgEGCCqGSM49AwEHA8IAB16X5Zn5
H1nk4c/bzLHq786csF0F1WqLU8az8Po8ZLemh72xZHkPzNbg32qRELUbkG5cAY
V2HEM8QQLgUKLCgADAKBgqhkkjOPQQDAgNIADBFA1EA+eqNj8KPKVhd+J8kamDx
Gsumzj74uvuB4nakrL1/41CIDH/wgUTznbMgaKchg+lv8WY18xxTH7No4Tn1+Ix
/411
-----END CERTIFICATE REQUEST-----
```

[Back to CSR List](#)

CSR Details

- **Name, Key, and Profile**
Basic information about how the CSR was generated.

Profile Details

If the CSR was created using a profile, the page shows:

- **Template Name**
- **Profile Configuration Content** — including [req] and DN sections, if present.

This allows the user to review the exact settings used to generate the CSR.

Creation Time



Displays when the CSR was created.

CSR (PEM)

A read-only block containing the full PEM-encoded CSR:

```
-----BEGIN CERTIFICATE REQUEST-----
```

```
...
```

```
-----END CERTIFICATE REQUEST-----
```

This can be copied for external tools or manual signing.

Navigation

- **Back to CSR List** returns to the main CSR table.

Generate new CSR

The **Generate a New CSR** page lets users create a certificate signing request based on an existing key and certificate template.



Generate a New CSR

CSR Name:

New CSR

Select Key:

6wind (EC)

Select Certificate Template:

https-test

Certificate Template Content Preview:

```
[ req ]
prompt          = no
distinguished_name = dn
default_md       = sha256
default_bits     = 2048
default_ec_curve  = prime256v1

[ dn ]
C = IL
ST = IL
L = TLV
O = RAD
OU = RND
CN = https-test.csrname.test
```

Generate CSR

[Back to CSR List](#)

Fields

- **CSR Name**
Friendly name for the CSR (shown later in the CSR list).
- **Select Key**
Choose an existing key (RSA, EC, or PQC) that will be used to build the request.
- **Select Certificate Template**
Select a certificate template that defines the DN and other CSR parameters.
- **Certificate Template Content Preview**
Read-only preview of the selected template (for example, [req] and DN sections) so the user can verify the configuration before generating the CSR.

Actions

- **Generate CSR**
Creates the CSR and returns to the CSR list, where the new request is available for viewing, downloading, or signing.



- **Back to CSR List**
Cancels the operation and returns to the list without creating a CSR.

3.9 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

Chose server extension , either None , User or System one.

Pikachu CA (R&D Only)

Submit a CSR

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBEzCBugIBADBYMQswCQYDVQQGEwJJUzERMA8GA1UECAwIVGVsX0F2aXYxDDAK
BgNVBACMA1RMVjEMMAoGA1UECgwDUkFEMQwwCgYDVVQLDANSTkQxDDAKBgNVBAMM
A1NTTDBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABi6X5Zn5H1nkk4c/bzLHq786
csfOF1WqLU0az0Po8ZLemh72xZHHkpZnBg32qRELUBkG5cAYV2HEW8QCQLgUKLCg
ADAKBggqhkjOPQDAgNIADBFaiBR1Nlb/L4GiDDLzB6ZO3DMLGUzUf1oLCOqDj7Y
AenZNglhAM207xLBuxL4hdG3if4CVS5p1HVUsV3yZ0xvarCS+Y0w
-----END CERTIFICATE REQUEST-----
```

Sign CSR

Clear CSR

Load a Pending CSR Request

HTTPS (Created: 2025-06-18 14:36:01.305540)



Load CSR into Form

X509 Extensions

None

None

System

User

Manage X509 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
```

Curl

send the enrolment command using curl

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

For mTLS tests use the following curl

```
curl --cert https.crt --key https.key -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
```

Estclient

enroll using command estclient

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

mTLS enroll using command estclient

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

SCEP

For SCEP you must provide the CA of either full chain (Root + Sub CA) or only the intimate one (Sub CA)
send the enrolment command using scep command (see Complementary section for installing it)

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

Note: to debug scep 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 \  
http://openxpki.iot-rad.com/.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>



3.10 Verification Authority (VA)

The Verification Authority page displays all certificates that have been revoked by the CA and provides access to the generated Certificate Revocation List (CRL).

List of Revoked Certificates

Verification Authority (VA)

List of Revoked Certificates

ID	Subject	Serial
1	18-06-F5-B7-92-D5	0x43d5fd07c3e6a23a
3	ABIGAIL-N-Pikachu-2	0x7510f27b2569d378
4	ABIGAIL-N-Pikachu-3	0xfebc9d3f088e437b
5	6WIND	0x22d7fd7e95466f5e
8	YOSSI-N-Pikachu-1	0x282b46572f9d55f2
64	sscep-test-22:41:8:12:2025	0x2e053f2ad6fd708a0f4792cb7de14b7a89f3bca5
65	estclient-go-22:47:8:12:2025	0x39417034a598d307f950eddd90169581bc4816e3
69	estclient-go-13:42:9:12:2025	0x6f0458f1c2a34b274e2c882a7976a8cca79b4f0e
96	https-est-secure-test	0x965fcb3b4a781339

[Download CRL](#)

Raw CRL Output (OpenSSL)

```
Certificate Revocation List (CRL):
Version 2 (0x1)
Signature Algorithm: sha256WithRSAEncryption
Issuer: C=IL, ST=TLV, L=Tel Aviv, O=RAD, OU=RD, CN=RADSubTestRSA
Last Update: Dec 12 14:49:25 2025 GMT
Next Update: Dec 19 14:49:25 2025 GMT
Revoked Certificates:
Serial Number: 43D5FD07C3E6A23A
Revocation Date: Dec 12 14:49:25 2025 GMT
Serial Number: 7510F27B2569D378
Revocation Date: Dec 12 14:49:25 2025 GMT
Serial Number: FEBBC9D3F088E437B
Revocation Date: Dec 12 14:49:25 2025 GMT
Serial Number: 22D7FD7E95466F5E
Revocation Date: Dec 12 14:49:25 2025 GMT
```

This table shows all revoked certificates, including:



- **ID**
Internal index for reference.
- **Subject**
The certificate's Common Name (CN) or identifying label.
- **Serial**
The serial number of the revoked certificate, used for validation and OCSP/CRL checks.

This list reflects the current content of the active CRL and updates whenever a certificate is revoked.

Download CRL

The **Download CRL** button provides the complete CRL file in PEM format.

This file can be published or distributed to systems that need to validate certificate status.

Raw CRL Output (OpenSSL)

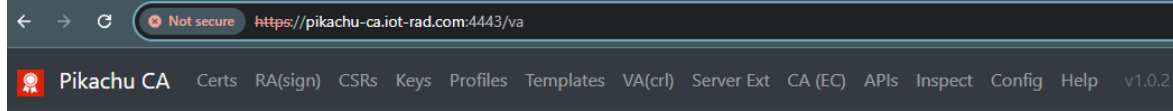
A raw, decoded view of the CRL is displayed below the table.

It includes:

- **CRL version and signature algorithm**
- **Issuer of the CRL**
- **Next Update** (when a new CRL will be required)
- **Revoked certificates with:**
 - Serial number
 - Revocation date

This output is generated using OpenSSL and is useful for troubleshooting and verifying correct CRL structure.

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 CRL](#)

3.11 Inspect

The **Inspect Data** tool allows users to paste any PEM block or Base64-encoded DER data and automatically with openssl command decode it into a readable structure.

Inspect Data

Paste PEM block or Base64-encoded DER:

[Inspect](#)

[Clear](#)

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



How It Works

- Paste a certificate, CSR, CRL, key, OCSP message, or PKCS file into the text box.
- Click **Inspect** to decode the content and display its parsed details.
- Click **Clear** to reset the input field.

Supported Types

The inspector can recognize and decode:

- Certificate Signing Request (CSR)
- X.509 Certificate
- Certificate Revocation List (CRL)
- PKCS#7 / CMS
- PKCS#12 / PFX
- OCSP Request
- OCSP Response
- Private Key
- Public Key

This tool simplifies validation and troubleshooting by providing quick access to decoded cryptographic objects without needing external utilities.

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:
```

3.12 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.

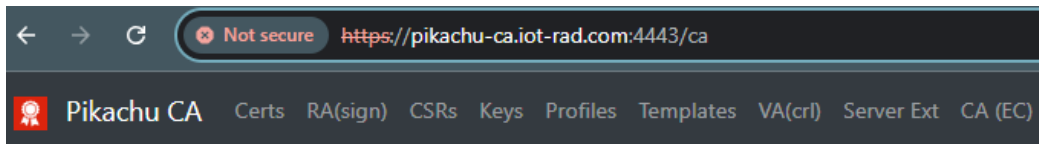
Sub CA (Intermediate) Certificate:

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.

3.13 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 http://openxpki.iot-rad.com/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 ocsf_response.der contains the status as well
command

```
openssl ocsf -respin ocsf_response.der -text -noverify
```

stdout

```
OCSP Response Data:
  OCSF Response Status: successful (0x0)
  Response Type: Basic OCSF 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
```

The support OCSF multi certificates checks in one request

Return OCSF response if certificate is unknown to the OCSF CA responder

Certificate based Servers using OCSF

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

OCSF 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.

3.14 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-----

```
MIIPQTCBbcCAQAweljELMAkGA1UEBhMCRIxJfjAUBgNVBAGMDUlsZS1kZS1GcmFu
Y2UxZjAMBgNVBAcMBVBhcmIzMQ4wDAYDVQQKDAU2V0lORDEUMBGA1UECwwLQ0Eg
RGI2aXNpb24xHTAbBgNVBAMMFdZXSU5EIFRlc3QgRUNEU0EgUknBIIIFMjALBglg
hkgBZQMEAxEDggUHAEEpVUmVR+P7L8A9sb53b/ua/oLtB0/Q1IFk2k9l8UILL1sa
8cogEKPo4GhPSG8UM5M4Uk457zfbPawslzXYR3zHOkRmwaxazYSj5o6UofVopnjh
/TALCEXvaDvhYdCvVRFDVZBI4MGH+MPHdpwmysXXXf59GmOXztp7KVXzP9XzYW
xxb7dva3NGpDOR5zEMmClxPEyabqd0oMnhJX2/k2j0j41Hmd+ZgPs6Hl/XPkJWI
gSJWgbKg7Tbct4AkuZ5EcafW1CfDxgbcVQABPvsCqrXBSd3XX1+CHom3Pu1t+ntM
ep1XKtRfJl3EkWl0CiDkOeV5gWjU/cvacV2v6VbnCR+vFWDdGJEUjyiA8q3mo7
```

Sign CSR

Clear CSR

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 below

Issued Certificates

Filter by Common Name, Serial, Key or Date

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

3.15 APIs

List of supported server APIs



General

Server APIs & CLI Commands

Type	Action	Endpoint URL / Command	Misc
CA General	Download CA Chain	<code>curl -k https://pikachu-ca.iot-rad.com:5443/downloads/chain</code>	Returns full CA chain file
	Download CRL	<code>curl -k https://pikachu-ca.iot-rad.com:4443/downloads/cr1</code>	Generates and downloads latest Certificate Revocation List
	Certificate Status	<code>curl -k https://pikachu-ca.iot-rad.com:4443/status/0xc75f573d9cb2b581</code>	Returns status as valid, revoked, or not found in JSON
	Expired Certificates	<code>curl -k https://pikachu-ca.iot-rad.com:5443/expired</code>	Returns list of certificate IDs that are expired
	Download CSR	<code>curl -k https://pikachu-ca.iot-rad.com:5443/requests/1/download</code>	Serves saved CSR if available

SCEP

SCEP	SCEP CRL	<code>sscep getcrl \ -u http://pikachu-ca.iot-rad.com:8090/scep \ -c ca_rsa.crt \ -k local.key \ -l local.crt \ -w crl.pem</code>	Not Supported
	SCEP Enrollment	<code>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</code>	prerequisite RSA based Sub CA @Work
	SCEP CA Certs	<code>sscep getca -u http://pikachu-ca.iot-rad.com:8090/scep -c cap.pem</code>	prerequisite RSA based Sub CA



EST

The EST APIs accessible via *curl* and *estclient*.

support operation over both insecure HTTPS endpoints and secure mTLS-protected endpoints.

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



OCSP

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

3.16 Inspect

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

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-----
MIIBXjCCAQQCAQEWcGyIKoZlZj0EAWlwYzELMAkGA1UEBhMCSUwxDDAKBgNVBAGM
A1RMTjEJERMA8GA1UEBwwlVGvsIEF2aXYxDDAKBgNVBAoMA1JBRDELMAkGA1UECwwC
UkQxGDAWBgNVBAMMD1JBRFN1YlRlc3RFQ0RTQRcNMjUwNDE2MTI0NTlyWhcNMjUw
NDIzMTI0NTlyWjBwMBoCCQDdHX1c9nLK1gRcNMjUwNDE2MTI0NTlyWjAaAgKA95tx
pGuBj3UXDTI1MDQxNjEyNDUyMlowGgIJAJY+W9jo+3ahFw0yNTA0MTYxMjQ1MjJa
MBoCCQD2+Bm0+ul8MRcNMjUwNDE2MTI0NTlyWjAKBgqqhkjOPQQDAgNIADBFaiAd
tsVuaWTXZjNzqWZ4alzeK8w1FV4kUj1/DNozDSvYZQIhAO2K4LHSZEHYjV9oUVuM
XWGFU4cZGO3ldB5PNhj06vBH
-----END X509 CRL-----
```

Inspect

Clear

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: C75F573D9C82B581
    Revocation Date: Apr 16 12:45:22 2025 GMT
  Serial Number: F79B71A46B818F75
    Revocation Date: Apr 16 12:45:22 2025 GMT
  Serial Number: 963E5BD8E8F876A1
    Revocation Date: Apr 16 12:45:22 2025 GMT
  Serial Number: F6F819B4FAF97C31
```

3.17 Config

Config Navigation Tab shows the server current configuration, secret protected

3.18 ChatPikachu

ChatPGT PGT related to the CA server features and capabilities



Pikachu CA 40 ▾

how to create PQC key using the pikachu UI

Pika Pi! To create a **Quantum-Safe (PQC) key** using the Pikachu UI for RAD's certificate management platform, follow these zappy steps:

⚡ Steps to Create a PQC Key via the Pikachu UI:

1. Access the UI:

Go to <https://openxpki.iot-rad.com:4443> ➤

2. Generate PQC Key:

- Navigate to the "Keys" section via management links or directly open: <https://openxpki.iot-rad.com:4443/keys>
- Click on "Generate New Key"
- Instead of choosing RSA or EC, **ensure you've installed and activated the `oqsprovider`** in OpenSSL (see prerequisites below).
- Use terminal (not UI) to generate a PQC key:

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

🔗 `mldsa44` = *Dilithium2* = *NIST Level 1 (quantum-safe)*

Other options: `mldsa65` (Level 3), `mldsa87` (Level 5)

3. Generate CSR:

- Create a config file `client1.cnf`:

ini

DEVELOPER MODE `.code`

+ Ask anything

3.19 Help

PDF format of this word document



PKI Hands-On 1.04

Last updated	19-Nov-2025
Doc. version	1.04
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	

3.20 Logs

The **Server Logs** page provides a live view of Pikachu CA server output, including warnings, errors, and operational messages.



Server Logs · Show last lines · every sec [Refresh](#) [Resume](#) · Filter [Clear Filter](#)

```
2025-12-04 19:29:36,547 WARNING [app] Failed to initialize CRL on startup: module 'datetime' has no attribute 'utcnow'
2025-12-04 19:29:36,582 INFO [werkzeug] WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
2025-12-04 19:29:36,582 INFO [werkzeug] WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
2025-12-04 19:29:36,583 INFO [werkzeug] WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
2025-12-04 19:29:36,583 INFO [werkzeug] WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
```

Features

- **Show last *N* lines**
Controls how many recent log lines are displayed.
- **Auto-refresh interval**
Refreshes the log display automatically every *N* seconds.
- **Refresh / Resume**
 - **Refresh** updates the log view immediately.
 - **Resume** re-enables automatic updates after pausing.
- **Filter**
Allows filtering log entries by keyword (e.g., warning, error, vault).
Clear Filter resets the view to show all log messages.

Log Output

The log window displays real-time server messages such as:

- Initialization warnings
- CA or CRL-related errors
- Flask/Werkzeug runtime notices
- Debug or informational events (based on configured log_level)

This tool is useful for troubleshooting enrollment issues (e.g. SCEP and EST), OCSP backend errors, Vault/LDAP connectivity problems, and general server health.

3.21 Account

The **Account** page allows the logged-in user to manage their own login credentials and session.



Account

Change Password
Current Password
<input type="password"/>
New Password
<input type="password"/>
Confirm New Password
<input type="password"/>
<input type="button" value="Change Password"/>

Logout
<input type="button" value="Logout"/>

Change Password

Update your account password by entering:

- **Current Password**
- **New Password**
- **Confirm New Password**

Click **Change Password** to apply the update.

Logout

Ends the current session and returns to the login page.

3.22 Users

The **Manage Users** page allows administrators to view, approve, suspend, modify, and remove user accounts. It supports both **local** and **LDAP** authentication sources.



Manage

Manage Users

[User Events](#)

Filter by any column

Show: 10 Showing 1-5 of 5

[← Previous](#) [Next →](#)

ID	Username	Email	Idle	Settings/Status	Actions
1	admin	admin@localhost	00:00	<div>Admin Local</div> <div>Active In</div>	(You)
6	pikauser2	pikauser2@rad.com		<div>User Local Active</div> <div>Out</div>	<div>Delete</div> <div>User</div> <div>Change Role</div> <div>Reset Password</div> <div>Suspend</div>
2	testing	uzi.golan@gmail.com		<div>User Local Active</div> <div>Out</div>	<div>Delete</div> <div>User</div> <div>Change Role</div> <div>Reset Password</div> <div>Suspend</div>
3	uzigolan	uzi.golan@gmail.com		<div>User Local</div> <div>Pending Out</div>	<div>Delete</div> <div>User</div> <div>Change Role</div> <div>Reset Password</div> <div>Approve</div> <div>Suspend</div>
5	yossizuk	yossi@rad.com		<div>User Local</div> <div>Pending Out</div>	<div>Delete</div> <div>User</div> <div>Change Role</div> <div>Reset Password</div> <div>Approve</div> <div>Suspend</div>

[Add User](#)

User Max Session idle time : 30m

Legend:

User Roles

Admin

 Admin

User

 User

Auth Source

LDAP

 LDAP

Local

 Local

User State

Active

 Active

Pending

 Pending

Suspended

 Suspended

Login Status

In

 Logged in

Out

 Logged out

Page Features

- **Filter Bar** – Search by username, email, role, state, or any visible column.
- **Show / Pagination** – Adjust the number of users displayed and navigate between pages.
- **User Events** – Opens the audit log of user activity.

Table Columns

- **ID**
Internal user ID.
- **Username**
The login name of the user.
- **Email**
User's email address.



- **Idle**
Idle time since last activity (based on configured max idle timeout).

Settings / Status

Displays several indicators:

- **Role:** Admin or User
- **Auth Source:** Local or LDAP
- **User State:** Active, Pending, Suspended
- **Login Status:** In (logged in) or Out (logged out)

Actions

Administrators may perform the following actions:

- **Delete** – Removes the user account.
- **Change Role** – Switch between *Admin* and *User*.
- **Reset Password** – Available for *Local* users only.
- **Approve** – Activates users with a *Pending* state.
- **Suspend** – Temporarily disables the user account.

1. Additional Options

- **Add User** – Creates a new local user account.
- **User Max Session Idle Time** – Displays the configured timeout for automatic logout.

The currently logged-in administrator sees **(You)** next to their own entry and cannot delete themselves.

User Events

The User Events page provides an audit trail of all user-related actions in the system.

It allows administrators to monitor logins, logouts, role changes, account updates, and enforcement actions such as suspensions or forced logouts.



User Events

Show: 10 Showing 1-10 of 20

[← Previous](#) [Next →](#)

Filter by User, Actor, Type, or Details

Time ↕	User ↕	Actor ↕	Event Type ↕	Details ↕
2025-12-12 15:25	testing	admin	suspend	
2025-12-12 15:22	mayagolan	admin	Delete	
2025-12-12 15:22	pikauser3	admin	Delete	
2025-12-12 13:05	admin	admin	login	ip: 127.0.0.1
2025-12-12 13:05	admin	admin	Force Logout	
2025-12-11 18:47	admin	admin	login	ip: 127.0.0.1
2025-12-11 18:37	admin	admin	Force Logout	
2025-12-11 16:16	admin	admin	login	ip: 127.0.0.1
2025-12-11 16:16	admin	admin	Force Logout	
2025-12-11 15:57	pikauser2	pikauser2	Force Logout	

Event Types Legend:

Add Add User
Delete Delete User
Pending Pending
Activated Activated
Deactivated Deactivated
Force Logout Force Logout
Change Password Change Password

Details
Additional info about the event (IP, reason, etc.)

Table Columns

- **Time**
Timestamp of when the event occurred.
- **User**
The account affected by the event.
- **Actor**
The user who performed the action
(*e.g., an admin who suspended another user*).
- **Event Type**
Categorizes the action, such as:
 - **login** – User successfully logged in
 - **Force Logout** – Admin forcibly logged out a user



- **Delete** – User account deleted
 - **suspend** – User temporarily disabled
 - **Change Password** – Password updated
 - **Activate / Pending / Deactivated** – Account state transitions
- **Details**
Extra information such as IP address or system notes.

Event Types Legend

- **Add** – User created
- **Delete** – User removed
- **Pending** – Awaiting approval
- **Activated** – User enabled
- **Deactivated** – User disabled
- **Force Logout** – Session terminated by admin
- **Change Password** – Password updated

Details

Shows additional context such as the source IP, actor name, or system-generated commentary.

Add User

The **Add User** page allows administrators to create new **local** user accounts.



Add User

Username

admin

Password

.....

Role

User

Email

UZI_G@RAD.COM

Add User

Cancel

2. Fields

- **Username**
The login name for the new account. Must be unique.
- **Password**
The user's login password. This applies only to local users (LDAP users authenticate externally).
- **Role**
Specifies the user's permission level:
 - **User** – Standard user privileges
 - **Admin** – Full administrative access, including user and CA management
- **Email**
Email address associated with the account. Used for identification and audit purposes.

3. Actions

- **Add User**
Creates the new account and redirects back to the Manage Users page.
- **Cancel**
Returns to the Manage Users page without creating a user.

Once added User is at pending state and waits for admin approval



Register User

The **Register** page allows new users to create a local account on the system.

Register

Username

Username must be at least 6 characters, max 20, and contain only letters, numbers, and underscores.

Email

A valid email is required.

Password

Password must be at least 6 characters.

Confirm Password

Passwords must match.

[Already have an account? Login](#)

Fields

- **Username**
Must be 6–20 characters long and contain only letters, numbers, and underscores.
- **Email**
A valid email address is required.
- **Password**
Must be at least 6 characters long.
- **Confirm Password**
Must match the password exactly.

Actions

- **Register**
Creates the user account if all fields are valid.
The account will appear in the *Manage Users* list, typically in a **Pending** state until approved by an administrator (unless auto-approval is enabled).



- **Login**
Link to the login page for existing users.

Appendixes

A.1. Complimentary openssl commands

Generate CSR

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

A.2. MQTTs

The MQTT broker **Mosquitto** is commonly used as a server that leverages the CRL file to deny access to clients presenting revoked certificates.

Configuration updates must be applied in **mosquitto.conf** (refer to Paragraph 2.2 *installation / Complementary / MQTT Broker*).

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

The MQTT server denies connection attempts made with revoked certificates.
To ensure proper enforcement, the server must be periodically updated with the latest CRL.

For example, a CRL updater service can be added to the **docker-compose** configuration to automatically fetch and refresh the CRL file.

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

A.3. APIs

The platform supports the following features either UI or API methods

Note all https commands are Server-only TLS unless stated mTLS (Mutual TLS) for Both server and client authenticate with certificates. (default mTLS port 4443)

CA General	Download CA Chain	curl -k https://pikachu-ca.iot-rad.com/downloads/chain	Returns full CA chain file
	Download CRL	curl -k https://pikachu-ca.iot-rad.com/downloads/crl	Generates and downloads latest Certificate Revocation List
	Certificate Status	curl -k https://pikachu-ca.iot-rad.com/status/0xc75f573d9cb2b581	Returns status as valid, revoked, or not found in JSON
	Expired Certificates	curl -k https://pikachu-ca.iot-rad.com/expired	Returns list of certificate IDs that are expired
	Download CSR	curl -k https://pikachu-ca.iot-rad.com/requests/1/download	Serves saved CSR if available
SCEP	SCEP CRL	sscep getcrl -d -u http://pikachu-ca.iot-rad.com/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/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/scep -c cap.pem	
EST	EST Enrollment	curl -k -X POST --data-binary @etx.csr.der https://openxpki.iot-rad.com/.well-known/est/simpleenroll -H "Content-Type: application/pkcs10" --output etx.crt.p7	Accepts DER CSR and returns signed certificate in PKCS#7



		estclient -server openxpki.iot-rad.com -insecure -cacerts ca-bundle.pem -key device.key -csr device.csr -out device.crt enroll	
	EST CA Certs	curl -k https://pikachu-ca.iot-rad.com/.well-known/est/cacerts --output chain.crt estclient -server openxpki.iot-rad.com -insecure cacerts -out ca-bundle.pem	Returns CA chain in PKCS#7 format
EST mTLS	EST Enrollment	curl --cert https.crt --key https.key -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 estclient -server openxpki.iot-rad.com -cert https.crt -key https.key -cacerts ca-bundle.pem -key device.key -csr device.csr -out device.crt enroll	Config.ini trusted_port variable 4443
	EST CA Certs	curl --cert https.crt --key https.key https://pikachu-ca.iot-rad.com:4443/.well-known/est/cacerts --output chain.crt estclient -server openxpki.iot-rad.com -cert https.crt -key https.key cacerts -out ca-bundle.pem	Config.ini trusted_port variable 4443
OCSP	OCSP Responder	openssl ocsp -reqout ocp_request.der -issuer rad_ca_sub.crt -cert valid.crt -url http://pikachu-ca.iot-rad.com/ocsp -resp_text -respout ocp_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/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
--	-----------------	---	----------------------------------



Change History

Date of Issue	Revision	Author	Responsible	Change Summary	Page Count
8-Apr-2025	1.0.0	Uzi G.		Document baseline	42
16-Apr-2025	1.0.1	Uzi G.		Update UI, CA details, Validity time manage	43
23-Apr-2025	1.0.2	Uzi G.		Update UI, New Navigate tabs. configuration based	52
18-Jun-2025	1.0.3	Uzi G.		Add manufacture certificate URL end point, Add generation of root and sub certificate procedure	57
19-Nov-2025	1.0.4	Uzi G.		OCSF tested with mocana stack, added secret protection for various commands, added PFX format, create profiles without templates by editing	64
12-Dec-2025	2.0.0	Uzi G.		Fix SCEP , add Multitenancy , Users management fix bugs	110