SHIELD DOCUMENT

Version 1.0.0

**AUTHOR:**

**Parthiban Nithyanantham,** Software Engineer from India. Can be reached using email-id n.parthibann@gmail.com.

**INTRODUCTION:**

Shield is a python-flask based micro service which is used to

* Create Self-Signed SSL certificates
* Create Organizational certificate chain / Trusted certificate chain
* Create Server certificates / licensing using certificates
* Create Keypair (private key and public key)

All these data will be stored in mongodb and served based on the request.

**Pre-requisite:**

* Mongodb

**Installation Instructions:**

1. Download / Clone the repository.
2. Move inside the repository folder and run the following command “pip install –r requirements.txt”
3. Update config file with mongo db details “shield.conf”
4. Start the application by running the “shield” file present in the bin directory (e.g python shield)

**API Details:**

1. **Create Certificate**

Method: POST

Endpoint: http://<ip>:<port>/v1/certificates

Headers: Content-Type: application/json

Request-body:

{

"common\_name": "parthi1",

"valid\_from": "2018-05-21T22:22:26",

"valid\_till": "2019-08-20T22:22:26",

"cert\_type": "ca\_root",

"path\_length": 0,

"country": "IN",

"state": "Tamilnadu",

"locality": "chennai",

"organization\_name": "companyX",

"organization\_unit\_name": "branchX",

"subject\_alternate\_name": {"dns": "localhost,parthitest16", "ip\_address": "127.0.0.1"},

"signature\_algorithm": "sha256"

}

**Request Parameter Details:**

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Required / Optional** | **Details** |
| common\_name | Required | Name of the certificate |
| valid\_from | Required | Starting validity of the certificate |
| valid\_till | Required | Ending validity of the certificate |
| cert\_type | Required | Type of the certificate, can be any of the following (ca\_root, ca\_intermediate, end\_entity, self\_signed) |
| path\_length | Optional | No of intermediate ca certificates that can exist |
| Country | Optional | Country name |
| State | Optional | State name |
| Locality | Optional | Locality name |
| organization\_name | Optional | Organization name |
| organization\_unit\_name | Optional | Organization unit name |
| subject\_alternate\_name | Optional | Subject alternate name  Type: dictionary   * dns values are comma separated string * ip\_address values are comma separated string |
| signature\_algorithm | Optional | Signature algorithm, can be any of the following (sha1 / sha256) |
| issuer\_id | Optional (Required when cert\_type is ca\_intermediate or end\_entity) | Objectid of the issuer certificate |

Sample Response:

{

"status": "success",

"message": "Certificate created successfully.",

"id": "5b0accf183278510746312d3"

}

1. **View Certificate**

Method: GET

Endpoint: http://<ip>:<port>/v1/certificates/<certid>

Headers: Content-Type: application/json

Sample Response:

{

"status": "success",

"message": "Certificate information",

"data": {

"status": "active",

"certificate\_type": "ca\_root",

"valid\_till": "2019-08-20 22:22:26",

"valid\_from": "2018-05-21 22:22:26",

"path\_length": 0,

"country": "IN",

"created\_at": "2018-05-27 15:21:21.737000",

"locality": "chennai",

"organization\_name": "companyX",

"issuer\_id": null,

"signature\_algorithm": "sha256",

"common\_name": "parthi1",

"id": "5b0accf183278510746312d3",

"organization\_unit\_name": "branchX"

}

}

1. **List Certificate**

Method: GET

Endpoint: http://<ip>:<port>/v1/certificates

Headers: Content-Type: application/json

Query params:

Limit: number of records to view

skip\_val: number of records to be skipped

Sample Response:

{

"status": "success",

"recordsFiltered": 1,

"draw": 1,

"recordsTotal": 1,

"message": "Certificates list",

"page\_count": 1,

"data": {

"certificates": [

{

"id": "5b57540d83278511f843a81c",

"common\_name": "my\_cert",

"cert\_type": "ca\_root",

"signature\_algorithm": "sha256",

"valid\_from": "2018-07-24 21:53:00",

"valid\_till": "2018-07-31 21:53:00",

"key\_id": "5b57540d83278511f843a819",

"issuer\_id": null

}

]

}

}

1. **Delete Certificate**

Delete certificate just change the state from ‘active’ to ‘in-active’, only soft delete happens and the actual data will still exist in the database as in-active.

Method: DELETE

Endpoint: http://<ip>:<port>/v1/certificates/<certid>

Headers: Content-Type: application/json

Sample Response:

{

"status": "success",

"message": "Certificate deleted successfully."

}

1. **Download Resource(certificate / private key / public key)**

Method: GET

Endpoint: http://<ip>:<port>/v1/certificates/download/<type>/<resource\_id>

Headers: Content-Type: application/json

Sample Response:

The requested resource will get downloaded as a pem file.

**INFO:** type will be any of the following (cert, private\_key, public\_key), and resource\_id will be the id of the type specified.

1. **Download Resource(certificate / private key / public key)**

Method: GET

Endpoint: http://<ip>:<port>/v1/certificates?action=calist

Headers: Content-Type: application/json

Sample Response:

{

"ca\_root": [],

"self\_signed": [],

"ca\_intermediate": [

{

"id": "5b57540d83278511f843a81c",

"name": "my\_cert"

}

],

"end\_entity": [

{

"id": "5b57540d83278511f843a81c",

"name": "my\_cert"

}

]

}

**USAGE:**

* **CREATE SELF-SIGNED CERTIFICATE**

Using create certificate api, create a self-signed ssl certificate. Sample request body given below:

{

"common\_name": "parthi\_selfsigned",

"valid\_from": "2018-05-21T22:22:26",

"valid\_till": "2019-08-20T22:22:26",

"cert\_type": "self\_signed",

"path\_length": 0,

"country": "IN",

"state": "Tamilnadu",

"locality": "chennai",

"organization\_name": "companyX",

"organization\_unit\_name": "branchX",

"subject\_alternate\_name": "companyx.com",

"signature\_algorithm": "sha256"

}

The response will contain the id of the certificate created. Then, using download resources api, download the certificate pem file

* **CREATE TRUSTED CERTIFICATE CHAIN**

STEP 1: Using create certificate api, create CA root certificate, sample request body given below.

{

"common\_name": "parthiban\_caroot",

"valid\_from": "2018-05-21T22:22:26",

"valid\_till": "2019-08-20T22:22:26",

"cert\_type": "ca\_root",

"path\_length": 2,

"country": "IN",

"state": "Tamilnadu",

"locality": "chennai",

"organization\_name": "companyX",

"organization\_unit\_name": "branchX",

"subject\_alternate\_name": "companyx.com",

"signature\_algorithm": "sha256"

}

Response will contain the certificate id.

STEP2: Create ca intermediate certificate, same request body given below. Use above generated certificate id as issuer id.

{

"common\_name": "parthiban\_cainter",

"valid\_from": "2018-05-21T22:22:26",

"valid\_till": "2019-08-20T22:22:26",

"cert\_type": "ca\_inter",

"path\_length": 2,

"country": "IN",

"state": "Tamilnadu",

"locality": "chennai",

"organization\_name": "companyX",

"organization\_unit\_name": "branchX",

"subject\_alternate\_name": "companyx.com",

"signature\_algorithm": "sha256",

“issuer\_id”: “5b0adc828327853758dabf81”

}

Response will contain the certificate id.

STEP3: Create end\_entity / leaf / ssl certificate, sample request body given below. Use above generated certificate id as issuer id.

{

"common\_name": "parthiban\_endentity ",

"valid\_from": "2018-05-21T22:22:26",

"valid\_till": "2019-08-20T22:22:26",

"cert\_type": "end\_entity",

"path\_length": 0,

"country": "IN",

"state": "Tamilnadu",

"locality": "chennai",

"organization\_name": "companyX",

"organization\_unit\_name": "branchX",

"subject\_alternate\_name": "companyx.com",

"signature\_algorithm": "sha256",

“issuer\_id”: “5b0adcbf8327853758dabf8f”

}

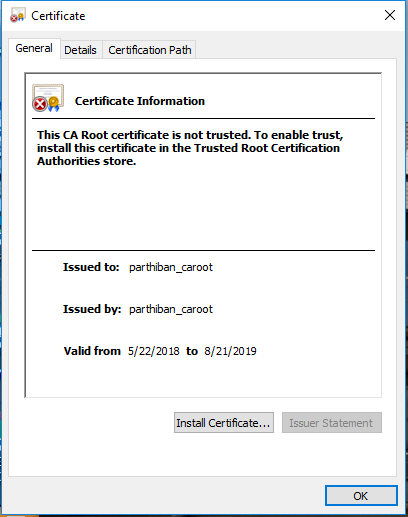
Response will contain the certificate id.

**ACCEPTANCE TESTING:**

STEP1: We have created an organization certificate chain above, download all the 3 certificates pem file using download resource api.

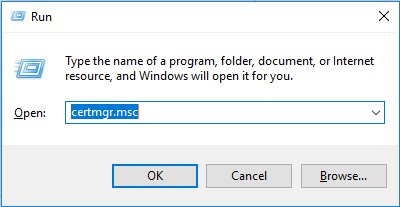
STEP2: Change the file extension from .pem to .crt and then open the certificate.

STEP3: When you open the certificate, you get the following information “**This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.**”

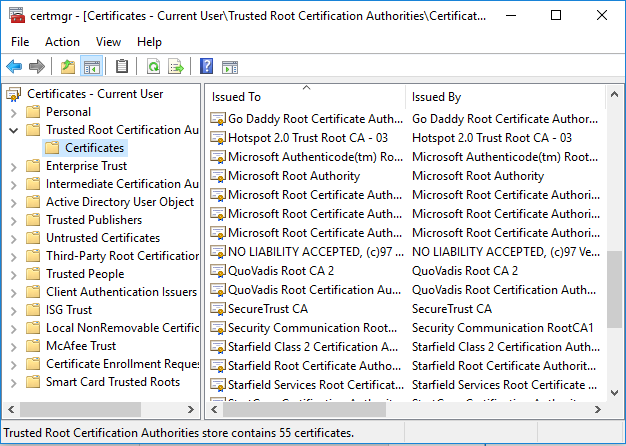


**Steps to install certificate in Trusted Root Certification Authorities store (windows):**

1. Open run prompt and type “certmgr.msc” and press enter.

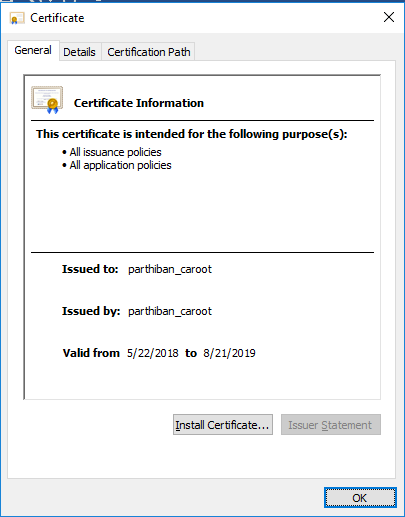


1. In “certmgr” window, select -> Trusted Root Certification Authorities -> certificates

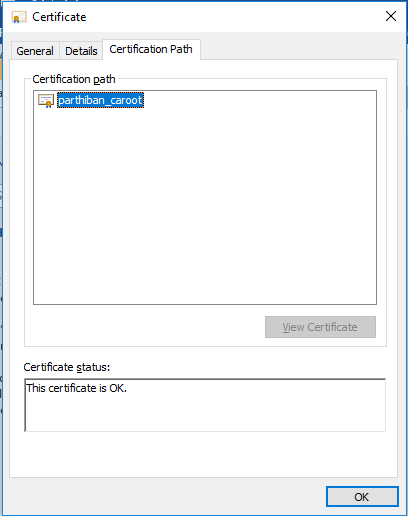


1. Right click on Certificates -> All Tasks -> Import and then import the root certificate.

STEP4: Once the import is successful, open the certificate again, this time you will see the intention of the certificate message.

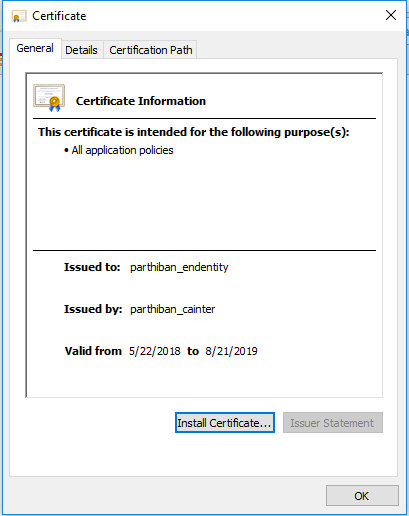
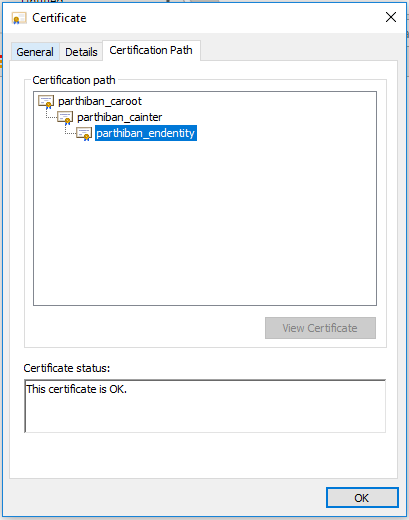


STEP5: Also navigate to “Certification Path” tab in the certificate and check the Certification Path and Certification status:



STEP6: Likewise, install the intermediate certificate in “Intermediate Certificate Authorities” store.

Step7: Finally open the end-entity certificate, and check for the certification path and certification status, if everything is ok then the certificate is valid for usage.

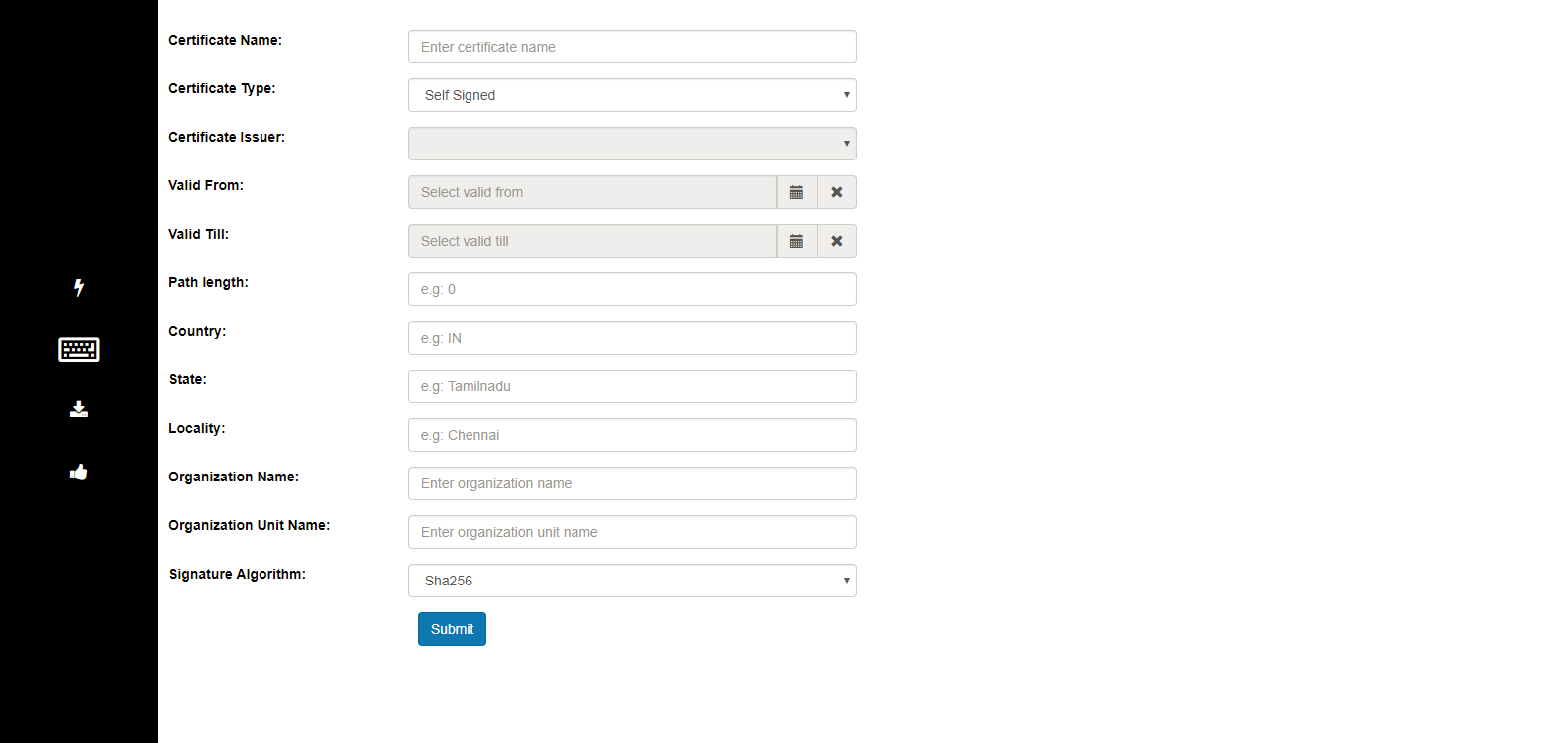


UI:

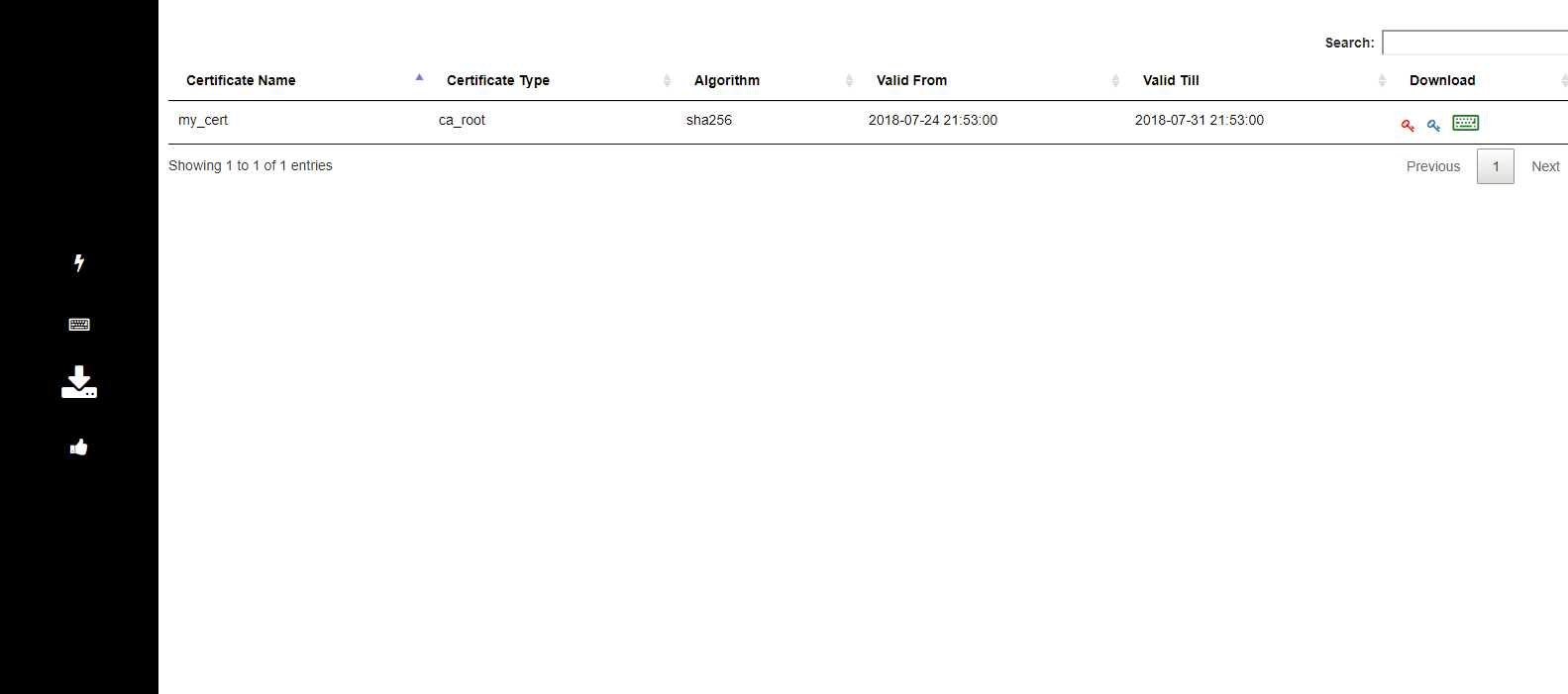
Landing page:



Certificate creation page:



Certificate downloads page:



Thank you:

