

Certificate and Credential Management in CORD

[Design Goals](#)

[Requirements](#)

[Definitions](#)

[Certificate Hierarchy](#)

[Challenges](#)

[Certificates during Development](#)

[Certificates in Deployment Scenarios](#)

[Public-facing API endpoints](#)

[Current Implementation](#)

Design Goals

Requirements

- All API endpoints must be secured using TLS
- All access to API endpoints must validate those SSL certificates
- The Certificate Revocation List (CRL) must be checked when validating certificates
- API endpoint must allow for their certificates to be updated/renewed in a way that doesn't cause major interruption to the operation of an installed system.

Certificate Hierarchy

CORD will create it's own internal-use SSL Certificate Authority (CA), with a 3-level hierarchy, where each layer signs only the certificates of the type below it:

1. Root CA
2. Intermediate CA
3. Server and Client Certs

The Root CA certificate will be distributed to all endpoints. A Server or Client cert will need to include the Intermediate Cert in its certificate chain for it to be validated.

Challenges

Current Ubuntu 14.04 is running an old version of Apache (2.4.7) which doesn't allow intermediate certificates in the same file as a server/client certificate, which breaks a variety of hierarchy-oriented workflows when using Juju to set up Keystone and other services. Updating to a newer version of Ubuntu, or moving away from Juju may help this situation.

Until then, the solution is to include Intermediate CA certs with the root store. This isn't a good solution, as every system needs every Intermediate CA cert installed, rather than just the Root CA cert, which is a deployment hassle.

Certificates during Development

During development, reasonable for developers to share their `platform-install/pki` folder, or for one developer to create a Root CA and then generate Intermediate CA certs for the other developers, then distribute them.

Certificates in Deployment Scenarios

In a live deployment, much care must be taken to secure the Root CA. It's recommended that the Root CA be generated on an offline machine, and used only to generate Intermediate CA certificates.

Having an Intermediate CA for each deployment area or person deploying - for example, one CA per site in a multi-site deployment, is recommended best practice.

Public-facing API endpoints

In production deployments, public-facing endpoints will need to be able to be trusted by someone without the CORD Root Cert in their list of trusted certs.

For this reason, there should be provisions to use a certificate trusted by an outside CA, or to provide the outside party with the CORD Root Cert.

Current Implementation

The current implementation for the generation of certificates is in `platform-install`, specifically the `pki-*` roles.

If you want to generate an SSL certificate for your service, add the information about the endpoint to the `server\_certs` data structure for the profile you're working on:

<https://github.com/opencord/platform-install/blob/master/vars/cord.yml#L26-L32>

```
server_certs:
  - cn: "servername.cord.lab"
    subj: "/C=US/ST=California/L=Menlo Park/O=ON.Lab/OU=Test
Deployment/CN=servername.cord.lab"
    altnames:
      - "DNS:servername.cord.lab"
      - "DNS:www.servername.cord.lab"
```

Description of these variables:

- `cn` is "Common Name", used as the filename of the generated files. Is ignored for server identification with newer implementations of SSL (see `altnames` below)
- `subj` is the "Subject" line of the certificate. It should contain a Distinguished Name for the certificate. Some guidance on naming is available here:
https://github.com/opencord/platform-install/blob/master/roles/pki-root-ca/templates/open_ssl_root.cnf.j2#L52-L68
- `altnames` - The Subject Alternative Names for the certificate. A list of all DNS names used to access this service.

Include the `pki-root`, `pki-intermediate` and `pki-cert` roles to generate Root/Intermediate CAs and the server certificates. All of the generation of keys and certs is done locally and is stored in the `/pki` directory in `platform-install`, then copied over to the target (head node) machine.

The `pki-install` role will add the CA certs to the system list of trusted certs on a target machine (the one rebuilt with `update-ca-certificates` in Ubuntu), and subsidiary service containers (this is more specific to the current CORD design than the other fairly generic `pki-\*` roles).

Once a certificate has been generated, to copy a server key/cert to a target machine use these paths:

- Key: `{{ playbook\_dir }}/pki/intermediate\_ca/private/{{ cn }}\_key.pem`
- Cert w/Intermediate CA chain attached: `{{ playbook\_dir }}/pki/intermediate\_ca/certs/{{ cn }}\_cert\_chain.pem`
- Cert (use this only if the service doesn't support the Intermediate CA chain): `{{ playbook\_dir }}/pki/intermediate\_ca/certs/{{ cn }}\_cert.pem`

An example of including certificates in a configuration file and base64 encoding them:

https://github.com/opencord/platform-install/blob/master/roles/juju-setup/templates/cord_juju_config.yml.j2#L19-L21