Entrust Validation Authority

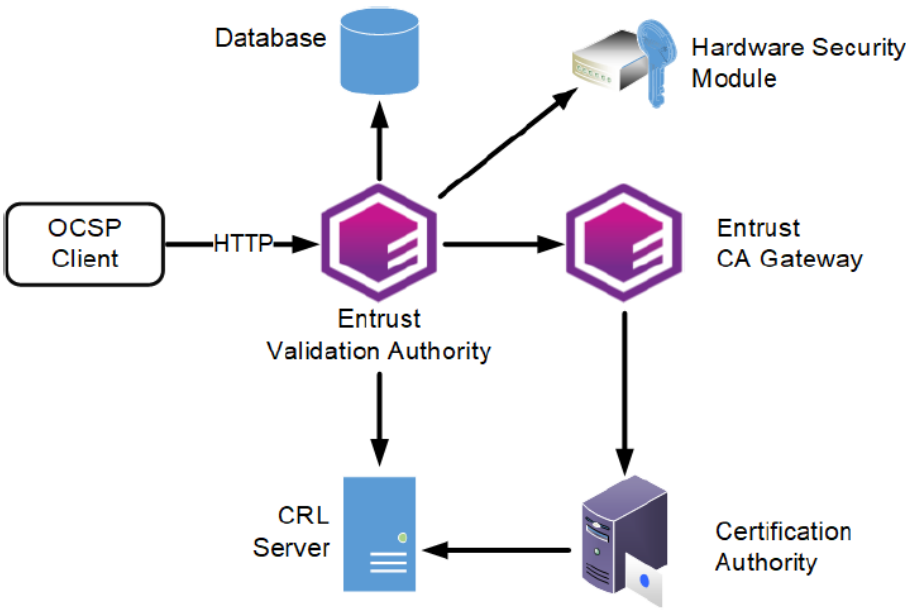
nShield® HSM Integration Guide

2026-02-10

Table of Contents

Chapter 1. Introduction

The Entrust Validation Authority (EVA) Server is an Online Certificate Status Protocol (OCSP) server for distribution of certificate revocation information for certificates issued by any certification authority (CA). The EVA Server provides integrity and validity for online transactions by validating, in real-time, digital certificates issued by a CA. The Entrust nShield Hardware Security Module (HSM) integrates with the Entrust Validation Authority server through the nShield PKCS #11 cryptography API to securely generate and store the OCSP response signing keys. To respond to OCSP requests, Entrust Validation Authority connects with different components.



In this architecture:

- Multiple clients send OCSP requests to the OCSP Responder service of Entrust Validation Authority.
- Multiple Certification Authorities (CAs) issue certificates.
- A Hardware Security Module (HSMs) managing one or several OCSP signing keys.
- One database stores the status of the certificates. For each CA, Entrust Validation Authority obtains the certificate status from either:
  - An Entrust CA Gateway instance.
  - A full or "combined" CRL published in an LDAP or HTTP server. Entrust Validation Authority does not support partitioned CRLs.

In this guide, the CA GW was not used and instead Entrust Validation Authority was configured using a Certificate Revocation List (CRL) published on a HTTP server.

1.1. Post-Quantum Ready

This integration brings together the power of Entrust’s next-generation nShield HSM platform and Entrust Validation Authority technology to deliver security built for the quantum era. With support for NIST-selected post-quantum algorithms, the combined solution gives organizations a confident, future-proof path to quantum-safe cryptography. By enabling seamless adoption of hybrid and quantum-resistant protection today, Entrust empowers customers to stay ahead of evolving threats and safeguard mission-critical keys and operations.

1.2. Requirements

The Entrust Validation Authority requires the following software:

- Entrust Deployment Manager 2.0.2
- Database
- Entrust CA Gateway or a CRL Server hosted on a HTTP server.
- Serial Number server (Not used in this integration)
- Certification Authority (CA)

Reference the *Entrust Validation Authority Deployment Guide* for product specific requirements.

Before starting this integration, review:

- The documentation for the nShield Connect HSM.
- The documentation and configuration process for Entrust Deployment Manager.
- The documentation and configuration process for Entrust Validation Authority.

Before using nShield products:

- When creating a Security World, identify custodians of the administrator card set (ACS).
- Obtain enough blank smart cards to create the ACS.
- Define the Security World parameters. For details of the security implications of the choices, see the *nShield Security Manual*.

Entrust recommends that you allow only unprivileged connections unless you are performing administrative tasks.

1.3. Licensing

Configuring Entrust Validation Authority requires importing a license text file into the Entrust Deployment Manager server administration web UI. Obtain the license file to configure Entrust Validation Authority in the Entrust Deployment Manager web UI. Also reference the *Entrust Validation Authority Deployment Guide* for product specific license requirements.

1.4. Product configurations

Entrust tested nShield HSM integration with Entrust Validation Authority in the following configurations:

Product	Version
Entrust Validation Authority	2.4.7
PKCS11 Library used in EVA	13.9.0-139-015f0
Entrust Deployment Management	2.0.2
Postgres Database	16
easy-rsa CA	3.1.7
Security World	13.9.3
HSM Hardware	Connect XC, nShield 5C

1.5. Supported features

Entrust tested nShield HSM integration with the following features:

Softcard	Module	OCS	nSaaS
Yes	Yes	Yes	Not Tested

1.6. Supported nShield hardware and software versions

Entrust tested with the following nShield hardware and software versions:

HSM	Security World Software	Firmware	Image
nShield 5c	13.9.3	13.8.4	13.9.3
nShield XC	13.9.3	13.8.3	13.9.3

1.7. Supported nShield functionality

Feature	Support
Key Generation	Yes
Key Management	Yes
FIPS 140 Level 3 mode support	Yes
Operator Card Set	Yes
Softcards	Yes
Module-only keys	Yes
Load Sharing	Yes

Chapter 2. Procedures

2.1. Download the Entrust software packages and documentation

Download the software files needed for the setup and installation.

1. Log in to https://trustedcare.entrust.com.
2. Go to **PRODUCTS** > **PKI** > **Entrust Validation Authority** and select the version that you want to download. (2.4.7)

   The **Entrust Validation Authority Page** appears.

3. From **Software Downloads**, download the Entrust Validation Authority files:
   - The `evactl` command-line tool.
   - The Entrust Validation Authority Installer (the solution file with the `sln` extension).
   - The `eva-config.json` sample configuration file.
   - The `eva-database-scripts.tar.gz` file that contains the database management scripts.
4. From **Documents**, download the *Entrust Validation Authority Deployment Guide*.
5. From the list of **Related Software**, select **Entrust Deployment Manager**.

   You will need it to install Entrust Validation Authority. This guide uses the VMware vSphere deployment. The version used in this guide for Entrust Deployment Manager is 2.0.2.

   Select the **Entrust Deployment Manager for VMware vSphere and physical machines** download option.

6. From **Documents** of the Entrust Deployment Manager, download the *Entrust Deployment Manager Installation and Administration Guide*.

2.2. Install and configure the database

Install and configure the database that will be used by Entrust Validation Authority. As explained in Entrust Validation Authority requirements, the Entrust Validation Authority uses an external database. To initialize this database, the `eva-database-scripts.tar.gz`, included with the software, provides scripts for each supported DBMS, which are:

- PostgreSQL database

- Oracle database
- SQL Server database

See the *Entrust Validation Authority Deployment Guide* for specific instructions on how to use the script to initialize the database.

This guide uses a server running the PostgreSQL database. An Ubuntu 24.04 server was deployed and PostgreSQL was selected to be installed. To install PostgreSQL on Ubuntu and configure it to be used by Entrust Validation Authority, do the following:

1. Install the `postgresql` package

   ```
   % sudo apt install postgresql
   ```

2. Allow other computers to connect to PostgreSQL database.

   Edit the `/etc/postgresql/16/main/postgresql.conf` file.

   ```
   % cd /etc/postgresql/16/main
   % vi postgresql.conf
   ```

   Locate the line:

   ```
   #listen_addresses = 'localhost'
   ```

   and change it to:

   ```
   listen_addresses = '*'
   ```

3. Set a password for the `postgres` user.

   Run the following command at a terminal prompt to connect to the default PostgreSQL template database:

   ```
   % sudo -u postgres psql template1
   ```

   The above command connects to PostgreSQL database `template1` as the `postgres` user. After you have connected to the PostgreSQL server, you will be at an SQL prompt. Run the following SQL command at the `psql` prompt to configure the password for the `postgres` user.

   ```
   ALTER USER postgres with encrypted password 'your_password';
   ```

Quit psql:

```
\q
```

4. Set the authentication type of the `postgres` user.

   Edit `/etc/postgresql/*/main/pg_hba.conf` to allow authentication with the `postgres` user from any system in the local network:

   ```
   host    all             all             x.x.x.1/24         trust
   ```

5. Restart the PostgreSQL service to initialize the new configuration.

   ```
   % sudo systemctl restart postgresql.service
   ```

6. Test the connection.

   ```
   % psql --host <db_host> --username postgres --password --dbname template1
   ```

7. Transfer the Entrust Validation Authority database scripts (`eva-database-scripts.tar.gz`) to the database server and `untar` the file.

   ```
   % tar zxvf eva-database-scripts.tar.gz
   % cd eva-database-scripts/postgresql
   ```

8. Create the environment variables needed to run the database scripts.

   The Values used here are the values used in the integration.

   DBNAME

   The database name.

   ```
   % export DBNAME=template1
   ```

   HOSTNAME

   The name of the host to connect to.

   ```
   % export HOSTNAME=<db_host>
   ```

   USERNAME

   The username to connect as.

```
% export USERNAME=postgres
```

PASSWORD

The password of the user to connect as.

```
% export PASSWORD='xxxxxx'
```

OCSPRESPONDER_DB_PASSWORD

The password of the OCSP Responder user with Read permissions on the `certStatus` and `metadata` tables.

```
% export OCSPRESPONDER_DB_PASSWORD='xxxxxxxxx'
```

OCSPRESPONDER_DB_USER

The name of the OCSP Responder user with Read permissions on the `certStatus` and `metadata` tables.

```
% export OCSPRESPONDER_DB_USER='ocspresponderuser'
```

STATUSFEEDER_DB_PASSWORD

The password of the Status Feeder user with Read and Write permissions on the `certStatus` and `metadata` tables.

```
% export STATUSFEEDER_DB_PASSWORD='xxxxxxxx'
```

STATUSFEEDER_DB_USER

The name of the Status Feeder user with Read and Write permissions on the `certStatus` and `metadata` tables.

```
% export STATUSFEEDER_DB_USER='statusfeederuser'
```

9. Run `certstatus_initial_schema.sql`.

```
% PGPASSWORD=$PASSWORD psql -d $DBNAME -U $USERNAME -h $HOSTNAME -v "ON_ERROR_STOP=1" -f
./certstatus_initial_schema.sql

CREATE TABLE
CREATE INDEX
```

10. Run `metadata_initial_schema.sql`.

```
% PGPASSWORD=$PASSWORD psql -d $DBNAME -U $USERNAME -h $HOSTNAME -v "ON_ERROR_STOP=1" -f
./metadata_initial_schema.sql

CREATE TABLE
CREATE INDEX
```

11. Run `create_users.sql` to create the database users.

```
% PGPASSWORD=$PASSWORD psql -d $DBNAME -U $USERNAME -h $HOSTNAME \
-v STATUSFEEDER_DB_USER=$STATUSFEEDER_DB_USER \
-v OCSPRESPONDER_DB_USER=$OCSPRESPONDER_DB_USER \
-v STATUSFEEDER_DB_PASSWORD=$STATUSFEEDER_DB_PASSWORD \
-v OCSPRESPONDER_DB_PASSWORD=$OCSPRESPONDER_DB_PASSWORD \
-v "ON_ERROR_STOP=1" -f ./create_users.sql

CREATE ROLE
CREATE ROLE
GRANT
GRANT
GRANT
GRANT
GRANT
```

2.3. Install and configure a Certificate Authority (CA)

Install and configure a Certificate Authority (CA) that will be used by Entrust Validation Authority. As explained in Entrust Validation Authority requirements, the Entrust Validation Authority uses a Certificate Authority (CA) to obtain the VA server certificate that will be used in the integration. Since the integration will not use a CA GW and instead it will use a Certificate Revocation List (CRL), the CA will also be used to generate the CRL list. The guide uses easy-RSA CA but any CA could be used in this case.

2.3.1. Install and configure a Certificate Authority Server

This guide uses the same server used for the database to install the CA. The installation instructions in this case are for an Ubuntu server. A separate server could be used if necessary.

1. Install the `easy-rsa` package.

```
% sudo apt install easy-rsa
```

2. Prepare the public Key Infrastructure directory.

```
% mkdir ~/easy-rsa
% ln -s /usr/share/easy-rsa/* ~/easy-rsa/
% chmod 700 ~/easy-rsa
% cd ~/easy-rsa
% ./easyrsa init-pki

init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: ~/easy-rsa/pki
```

3. Create a Certificate Authority.

   a. Create the `vars` file.

      Before you can create your CA's private key and certificate, you need to create and populate a `vars` file in the `easy-rsa` directory with some default values.

```
set_var EASYRSA_REQ_COUNTRY    "<country>"
set_var EASYRSA_REQ_PROVINCE   "<province>>"
set_var EASYRSA_REQ_CITY       "<city>"
set_var EASYRSA_REQ_ORG        "<organization>"
set_var EASYRSA_REQ_EMAIL      "xxxx.xxxx@entrust.com"
set_var EASYRSA_REQ_OU         "Interop"
set_var EASYRSA_ALGO           "ec"
set_var EASYRSA_DIGEST         "sha512"
```

   b. Create the root public and private key pair for your Certificate Authority.

```
% ./easyrsa build-ca

...
Enter New CA Key Passphrase:
Re-Enter New CA Key Passphrase:
...
Common Name (eg: your user, host, or server name) [Easy-RSA CA]: CA Gateway

CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
~/easy-rsa/pki/ca.crt
```

   c. Save the `ca.crt` file.

      This file will be used in the Entrust Deployment Management server to generate the keys. It will also be used in the Entrust Validation Authority GUI configuration.

2.3.2. Generate a server key pair

This key pair will be used to generate a certificate that will be revoked so we can generate the CRL that will be used by the integration. To generate the key pair, run the following command:

```
% keytool -genkeypair -alias <alias> -dname <dn> -keyalg <keyAlg> -keysize <keySize> \
-sigalg sha256WithRSA -ext san=dns:<dns> -keystore <keystore> [-keypass <keyPass>] [-storepass <keystorePass>]
```

For this integration:

```
% mkdir ~/cagw
% cd ~/cagw
% keytool -genkeypair -alias example_alias -dname "cn=CA Gateway,ou=CA Entry,o=Interop,c=US" -keyalg RSA -keysize
2048 -sigalg sha256WithRSA -ext san=dns:interop.local -keystore ./keystore.ks

Enter keystore password:
Re-enter new password:
Enter key password for <example_alias>
        (RETURN if same as keystore password):
Re-enter new password:
```

2.3.3. Obtain the key pair CSR

Create a Certificate Signing Request (CSR) by entering the following command:

```
% keytool -certreq -alias <alias> -file <file> -storetype pkcs12 -keystore <keystore> [-storepass <keystorePass>]
```

For this integration:

```
% keytool -certreq -alias example_alias -file ./cagw_csr.txt -keystore ./keystore.ks

Enter keystore password:
```

2.3.4. Obtain the server certificate from the CSR

1. Import the CSR

   Using the CSR generated in the previous step (`cagw_csr.txt`), import the CSR.

   ```
   % cd ~/easy-rsa
   % ./easyrsa import-req /tmp/cagw_csr.txt cagw

   Note: using Easy-RSA configuration from: ./vars

   Using SSL: openssl OpenSSL 1.1.1f  31 Mar 2020

   The request has been successfully imported with a short name of: cagw
   You may now use this name to perform signing operations on this request.
   ```

2. Sign the CSR.

   ```
   % ./easyrsa sign-req server cagw

   Note: using Easy-RSA configuration from: ./vars

   Using SSL: openssl OpenSSL 1.1.1f  31 Mar 2020


   You are about to sign the following certificate.
   Please check over the details shown below for accuracy. Note that this request
   has not been cryptographically verified. Please be sure it came from a trusted
   ```

```
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 1080 days:

subject=
    countryName               = US
    organizationName          = Interop
    organizationalUnitName    = CA Entry
    commonName                = CA Gateway


Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes
Using configuration from /home/xxxx/easy-rsa/pki/safessl-easyrsa.cnf
Enter pass phrase for /home/xxxx/easy-rsa/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'US'
organizationName      :PRINTABLE:'Interop'
organizationalUnitName:PRINTABLE:'CA Entry'
commonName            :PRINTABLE:'CA Gateway'
Certificate is to be certified until Oct  4 18:55:17 2026 GMT (1080 days)

Write out database with 1 new entries
Data Base Updated

Certificate created at: /home/xxxx/easy-rsa/pki/issued/cagw.crt
```

2.3.5. Revoke the certificate to be able to create the CRL list

To create the CRL needed for the integration, we need to revoke the certificate created in the previous step.

```
% ./easyrsa revoke cagw

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1f  31 Mar 2020


Please confirm you wish to revoke the certificate with the following subject:

subject=
    countryName               = US
    organizationName          = Interop
    organizationalUnitName    = CA Entry
    commonName                = CA Gateway


Type the word 'yes' to continue, or any other input to abort.
  Continue with revocation: yes
Using configuration from /home/xxxx/easy-rsa/pki/safessl-easyrsa.cnf
Enter pass phrase for /home/xxxx/easy-rsa/pki/private/ca.key:
Revoking Certificate 5FB65DF1FBD42CCA25FEC514B415E1BE.
Data Base Updated

IMPORTANT!!!

Revocation was successful. You must run gen-crl and upload a CRL to your
infrastructure in order to prevent the revoked cert from being accepted.
```

2.3.6. Generate the Certificate Revocation List

1. Generate the CRL.

   It will contain the certificate revoked in the previous step.

   ```
   % ./easyrsa gen-crl

   Note: using Easy-RSA configuration from: ./vars

   Using SSL: openssl OpenSSL 1.1.1f  31 Mar 2020
   Using configuration from /home/xxxx/easy-rsa/pki/safessl-easyrsa.cnf
   Enter pass phrase for /home/xxxx/easy-rsa/pki/private/ca.key:

   An updated CRL has been created.
   CRL file: /home/xxxx/easy-rsa/pki/crl.pem
   ```

2. Convert the `crl.pem` file to DER format

   Entrust Validation Authority expects the CRL to be in DER format.

   ```
   % openssl crl -in /home/xxxx/easy-rsa/pki/crl.pem -out /home/xxxx/easy-rsa/pki/crl.der -outform DER
   ```

3. Save the `crl.der` file so it can be made available in the HTTP host.

   The `crl.der` file needs to be made available via HTTP to Entrust Validation Authority.

2.3.7. Make the crl.der file available via http request

If a webserver is already available, use it and make the `crl.der` file available via that host. Otherwise, install Apache to make the `crl.der` file available via HTTP. In these instructions, we install Apache on the same Ubuntu server as the database server and CA server but it can be deployed on a separate server.

1. Install the `apache2` package.

   ```
   % sudo apt install apache2
   ```

2. Enable Apache so if the server is rebooted the Apache server runs.

   ```
   % sudo systemctl enable apache2
   ```

3. Start the Apache server.

   ```
   % sudo service apache2 start
   ```

4. Make the `crl.der` file available via URL in the Apache server.

   ```
   % cd /var/www/html
   % sudo mkdir crl
   % cp /home/xxxx/easy-rsa/pki/crl.der crl/.
   ```

   You should be able see the `crl.der` file now using the following URL:

   ```
   http://<apache_host>/crl/crl.der
   ```

2.4. Install and configure the nShield HSM

This guide does not cover the basic installation and configuration of the nShield HSM or the nShield Security World client software. For instructions, see the *Installation Guide* for your HSM.

Assuming Security World has been installed and configured and the World and modules have been created, prepare the `cknfastrc` file so it is setup according to the protection method selected for the integration. The file is in the `%NFAST_HOME%/kmdata` directory. `NFAST_HOME` is `C:\Program Files\nCipher\nfast` on Windows and `/opt/nfast` on Linux.

For more information about the environment variables used in `cknfastrc`, see:

- The *nShield Cryptography API Guide*.
- The PKCS #11 library environment variables section of the *User Guide* for the HSM.

2.4.1. Select the key protection method

2.4.1.1. Module protection

1. Add the following lines to the `cknfastrc` file of the Security World.

   ```
   CKNFAST_FAKE_ACCELERATOR_LOGIN=1
   ```

2. The token name used during module protection for the integration will be **accelerator**.
3. For FIPS 140 Level 3:
   a. You must have an OCS card created and inserted to provide FIPS-authentication.
   b. The ACS card can also be used to provide FIPS-authentication but it is not recommended.

2.4.1.2. Softcard protection

1. Add the following lines to the `cknfastrc` file of the Security World.

```
CKNFAST_LOADSHARING=1
```

2. Create a softcard

```
% ppmk -n testSC

Enter new pass phrase:
Enter new pass phrase again:
New softcard created: HKLTU 329333aa357af00ca57af28c3ca4a3b4e6d39afe
```

3. The token name used during softcard protection for the integration will be the softcard name used when you created the softcard. In this case, **testSC**.
4. For FIPS 140 Level 3:
   a. You must have an OCS card created and inserted to provide FIPS-authentication.
   b. The ACS card can also be used to provide FIPS-authentication but it is not recommended.

2.4.1.3. OCS protection

1. Add the following lines to the `cknfastrc` file of the Security World.

```
CKNFAST_LOADSHARING=1
```

2. Create the OCS card

```
% sudo /opt/nfast/bin/createocs -m1 -s2 --persist -N testOCS -Q 1/1

```

Creating Cardset:
 Module 1: 0 cards of 1 written
 Module 1 slot 0: Admin Card #1
 Module 1 slot 2: blank card
 Module 1 slot 3: empty
 Module 1 slot 4: empty
 Module 1 slot 5: empty
 Module 1 slot 2:- passphrase specified - writing card
Card writing complete.

cardset created; hkltu = a705fffe235cd68850ab08504622b233d7087d12
```

3. The token name used during OCS protection for the integration will be the name used when the OCS card was created. In this case, **testOCS**.
4. Insert the OCS card to provide FIPS-authorization.
```

2.4.2. cardlist file

If you are using a Remote administration card, add "*" to the `kmdata/config/cardlist` file to allow the usage of the remote admin card.

2.4.3. Create a tar file with the kmdata directory

When you are configuring Entrust Validation Authority, you will have to import the `kmdata` directory from the Security World into EVA.

1. Create a `tar` file that contains the `kmdata` directory.

   Make sure that the world, the modules, the cardlist, the softcard or the OCS cards, and the `cknfastrc` file have been created and that they are ready to be used by EVA.

   ```
   % cd /opt/nfast
   % tar czvf ~/kmdata.tgz kmdata
   ```

2. Save the `tar` file so it can be used when the import takes place.

2.5. Set up and configure the Entrust Deployment Manager server

Entrust Deployment Manager (EDM) provides a Management Console to deploy and manage Entrust solutions like Entrust Validation Authority. It needs to be setup and configure first before you can deploy Entrust Validation Authority. For this integration, EDM is deployed on VMware vSphere. After downloading the Entrust Deployment Manager software for vSphere. Install it according to what is documented in the EDM Administration Guide.

This integration used a VM with the following configuration:

- 4 CPUs
- 8 GB RAM
- 175 GB Disk

Once deployed, sign in using the following credentials:

- `sysadmin`/`changeme`

Follow the instructions on the Administration Guide to complete the installation. The EDM server was installed in single-mode for the integration.

```
% sudo clusterctl install --mode single-node
```

1. Replace the default password of the Management console.

   Open the following URL:

   ```
   https://<edm_host>/management-console
   ```

   a. Log in with the `admin` username and the `changeme` password.
   b. Fill in the Change Password form and select **SAVE**.

2. Replace the default Grafana password.

   Open the following URL:

   ```
   https://<edm_host>/grafana
   ```

   a. Log in with the `admin` username and the `changeme` password.
   b. Go to **Admin** > **Change Password** and change the admin's password.

2.6. Entrust Validation Authority setup and configuration

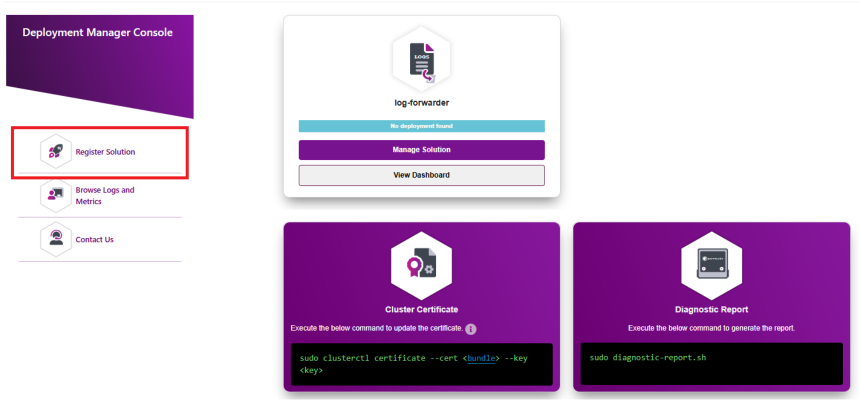
Entrust Deployment Manager provides a Management Console to deploy and manage Entrust solutions like Entrust Validation Authority.

Sign in to the **admin** account of the Entrust Deployment Management Console at the following URL:

```
https://<edm_host>/management-console
```

2.6.1. Register Entrust Validation Authority

1. Select **Register Solution** in the sidebar menu to display the registration dialog.



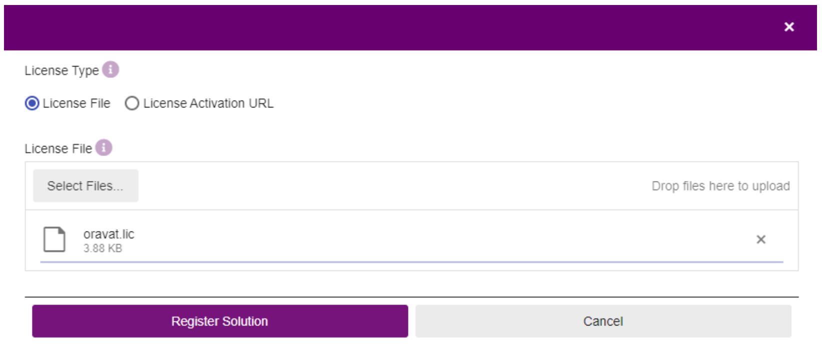
Execute the below command to update the certificate.

```
sudo clusterctl certificate --cert <bundle> --key
<key>
```

Execute the below command to generate the report.

```
sudo diagnostic-report.sh
```

2. In the registration dialog:
   a. Select **License File** to register the solution with a license file.
   b. Select **License Activation URL** to register the solution with an activation URL and a license password.



License Type

[✓ License File] License Activation URL

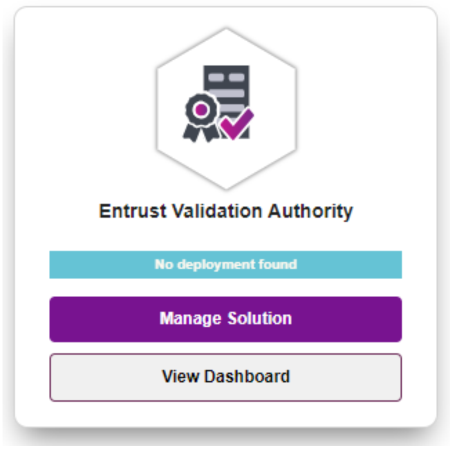
License File

oravat.lic
3.88 KB

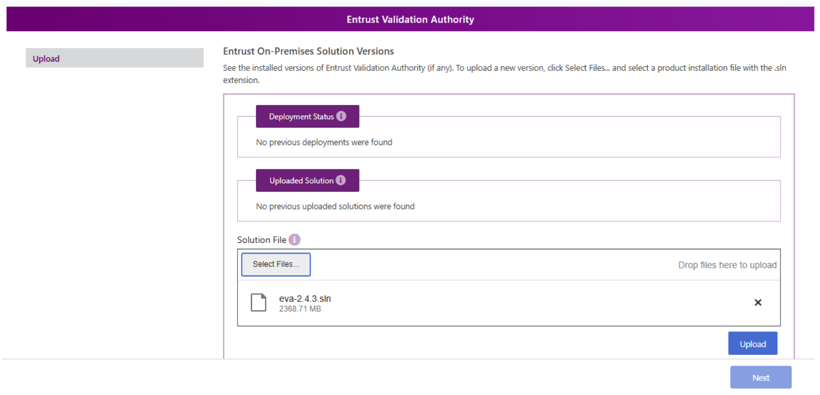
3. Select **Register Solution** and wait until the solution is registered.

2.6.2. Manage Entrust Validation Authority

1. In the content pane, select **Manage Solution** for the newly registered solution for **Entrust Validation Authority**.



2. To upload the solution file with the Management Console:
    a. In the **Entrust On-Premises Product Information** page, select **Select Files**.
    b. Select the `.sln` installation file for the solution.
    c. Select **Upload** and wait until the installation file is uploaded.
    d. Select **Next**.



Entrust On-Premises Solution Versions

See the installed versions of Entrust Validation Authority (if any). To upload a new version, click Select Files... and select a product installation file with the .sln extension.

No previous deployments were found

No previous uploaded solutions were found

eva-2.4.3.sln
2368.71 MB

2.6.3. Import the HSM configuration

1. As the `sysadmin` user, transfer the `kmdata.tgz` file created earlier with the HSM `kmdata` directory to the EDM server.

```
% sftp sysadmin@<edm_host>
> put kmdata.tgz
```

2. Transfer the `evactl` command that was downloaded with the EVA software downloaded to the EDM server.

```
% sftp sysadmin@<edm_host>
> put evactl
```

3. ssh to the EDM server IP address using the `sysadmin` user.
4. Extract the contents of the `kmdata.tgz` file.

```
% tar zxvf kmdata.tgz
```

Before importing the configuration, it is important to have the `cknfastrc` inside the kmdata folder. The file needs to be updated according to the type of HSM protection being used. See the Select the key protection method section for the contents of the `cknfastrc` file based on the protection method being used.

5. Run the `evactl import-nshield` command to import the configuration.

```
% sudo ./evactl import-nshield -f /home/sysadmin/kmdata

If there is a kmdata in EVA, it will be overwritten. Created keys will be lost. Continue? [y/N]: y
Uploading  done ┤|                                                      |├ 100 %
Secret(s) established. A redeploy of the eva solution is required for changes to take effect
Importing nShield...                       Done
```

2.6.4. Generate the VA certificate key and CSR

1. Generate the private key for the VA certificate.

   In the EDM server, run the `evactl create-key` command to generate a private key for the VA certificate. Do this based on the protection method selected in the HSM configuration section. Keep in mind the token name to be used.

   - If using softcard protection, change the token name to `testSC`.
   - If using module protection, change the token name to `accelerator`.
   - If using OCS protection, change the token name to `testOCS`.

     The command will also create the CSR needed for the VA certificate.

     The available types of keys are:

     - RSA2048
     - RSA3072
     - RSA4096
     - ECDSAP256
     - ECDSAP384
     - ECDSAP521
     - ML-DSA-44
     - ML-DSA-65
     - ML-DSA-87

     ```
     % sudo ./evactl create-key -k RSA2048 -s "CN=OCSP Server" -o /tmp/certreq.txt -t testSC -v nshield
     ```

     Here is an example of how to create it with a post-quantum key:

     ```
     % sudo ./evactl create-key -k ML-DSA-44 -s "CN=OCSP Server" -o /tmp/certreq.txt -t testSC -v nshield

     Obtaining necessary components for EVA...  Done
     Starting PKCS #11 Manager...               Done
     Using token with label testSC
     ```

```
Created key with id 5a870af47c7c908e6889ee98fb761e7fac78fb19
Setting  done ┤|                       |├ 100 %
Secret(s) established. A redeploy of the eva solution is required for changes to take effect
CSR written in path: /tmp/certreq.txt
```

2. Copy `/tmp/certreq.txt to /home/sysadmin/`.

```
% sudo chmod 777 /tmp/certreq.txt; cp /tmp/certreq.txt ~/.
```

3. Check that you can read the keys in the token.

```
% sudo ./evactl list-keys -t testSC -v nshield

Obtaining necessary components for EVA...  Done
Enter HSM PIN:
Starting PKCS #11 Manager...               Done
Using token with label testSC
Private Key Object; ML-DSA-44
  Label:      5a870af47c7c908e6889ee98fb761e7fac78fb19
  ID:         5a870af47c7c908e6889ee98fb761e7fac78fb19
  Usage:      sign

Public Key Object; ML-DSA-44
  Label:      5a870af47c7c908e6889ee98fb761e7fac78fb19
  ID:         5a870af47c7c908e6889ee98fb761e7fac78fb19
  Usage:      verify
```

2.6.5. Process the VA certificate request

Send the VA certificate request to a CA for generating a certificate with the following extension values:

Key Usage	`digitalSignature`
Extended Key Usage	`id-kp-OCSPSigning`

Extended Key Usage value in the documentation is `id-kp-OCSPSigning`. However, when we used `easy-rsa`, we had to set the value to `OCSPSigning` because using `id-kp-OCSPSigning` resulted in the following error message when the CRS was signed:

```
ERROR: adding extensions in section default
140384717350208:error:0D06407A:asn1 encoding routines:a2d_ASN1_OBJECT:first num too
large:../crypto/asn1/a_object.c:73:
140384717350208:error:2206706E:X509 V3 routines:v2i_EXTENDED_KEY_USAGE:invalid object
identifier:../crypto/x509v3/v3_extku.c:95:section:<NULL>,name:id-kp-OCSPSigning,value:<NULL>
140384717350208:error:22098080:X509 V3 routines:X509V3_EXT_nconf:error in
extension:../crypto/x509v3/v3_conf.c:47:name=extendedKeyUsage, value=id-kp-OCSPSigning

Easy-RSA error:

signing failed (openssl output above may have more detail)
```

Transfer the `certreq.txt` file to the CA server.

In the CA server, do the following:

1. Import the certificate

   ```
   % cd easy-rsa
   % ./easyrsa import-req /home/xxxx/certreq.txt va

   Using Easy-RSA 'vars' configuration:
   * /home/xxxxx/easy-rsa/vars

   Using SSL:
   * openssl OpenSSL 3.0.13 30 Jan 2024 (Library: OpenSSL 3.0.13 30 Jan 2024)

   Notice
   ------
   Request successfully imported with short-name: va
   This request is now ready to be signed.
   ```

2. Set the configuration of `easy-rsa` so the extension values are available.

   The `keyUsage` extension is already properly set. You only need to set the `extendedKeyUsage` extension. You do this by setting the following environment variable before signing the request:

   ```
   % export EASYRSA_EXTRA_EXTS='extendedKeyUsage = OCSPSigning'
   ```

3. Install an OpenSSL version that is compatible with post-quantum keys, if you are using them.

   If you are using a post-quantum key, you will need OpenSSL version 3.5.5 or higher.

   The following steps assume that you have OpenSSL installed and that it is installed in `/opt/openssl-3.5`.

   To use the new version of OpenSSL, you will have to setup the `PATH` and `LD_LIBRARY_PATH` environment variables:

   ```
   export PATH=/opt/openssl-3.5/bin:$PATH
   export LD_LIBRARY_PATH=/opt/openssl-3.5/lib64:$LD_LIBRARY_PATH
   ```

4. Sign the CSR.

   ```
   % ./easyrsa sign-req client va

   Using Easy-RSA 'vars' configuration:
   * /home/xxxxx/easy-rsa/vars

   Using SSL:
   * /opt/openssl-3.5/bin/openssl OpenSSL 3.5.5 27 Jan 2026 (Library: OpenSSL 3.5.5 27 Jan 2026)
   ```

```
You are about to sign the following certificate:
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.
Request subject, to be signed as a client certificate
for '825' days:

subject=
    commonName                = OCSP Server

Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes

Using configuration from /home/xxxxx/easy-rsa/pki/openssl-easyrsa.cnf
Enter pass phrase for /home/xxxxx/easy-rsa/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :PRINTABLE:'OCSP Server'
Certificate is to be certified until May  9 20:54:32 2028 GMT (825 days)

Write out database with 1 new entries
Database updated

Notice
------
Certificate created at:
* /home/xxxxx/easy-rsa/pki/issued/va.crt
```

5. Verify extensions are in the certificate

```
% openssl x509 -text -in /home/xxxx/easy-rsa/pki/issued/va.crt

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            87:23:49:6d:7f:8d:fd:42:1e:0c:33:5c:a0:51:c1:f9
        Signature Algorithm: ecdsa-with-SHA512
        Issuer: CN=CA Gateway
        Validity
            Not Before: Feb  4 20:54:32 2026 GMT
            Not After : May  9 20:54:32 2028 GMT
        Subject: CN=OCSP Server
        Subject Public Key Info:
            Public Key Algorithm: ML-DSA-44
                ML-DSA-44 Public-Key:
                pub:
                    c5:26:a2:af:db:a3:ef:b6:dc:1a:ae:14:89:d4:e8:
                    ...
                    ...
                    86:97:2f:84:f1:68:78
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                A8:08:A6:9D:44:88:CF:10:FF:0C:AB:4F:C9:F1:D7:F2:59:0B:1F:B1
            X509v3 Authority Key Identifier:
                keyid:5E:8C:5C:81:0A:BD:FA:3B:4E:E2:A6:8F:DC:68:B5:DE:D4:6C:AD:A6
                DirName:/CN=CA Gateway
                serial:98:67:7A:3E:52:BC:3D:65:0F:86:3B:D2:59:00:92:B1:31:E9:2D
            X509v3 Key Usage:
                Digital Signature
            X509v3 Extended Key Usage:
                OCSP Signing
    Signature Algorithm: ecdsa-with-SHA512
    Signature Value:
        30:65:02:30:46:97:70:10:bd:d5:71:68:a4:f6:11:8e:1a:ed:
        88:90:1f:e8:72:dc:fc:1e:3a:73:52:66:98:3b:dd:b7:70:38:
        3a:83:b2:3c:5b:46:86:a2:a9:ee:dc:5e:01:ca:c1:14:02:31:
        00:cf:1e:5e:0c:98:1f:0e:fe:0e:b9:13:53:2e:ea:9c:6c:ae:
        ff:48:48:c8:11:b6:92:dd:5b:79:b1:de:96:36:1b:a4:f4:ad:
        b9:33:bb:dd:e7:68:a3:36:4a:ef:a0:c8:1a
-----BEGIN CERTIFICATE-----
MIIGxzCCBk2gAwIBAgIRAIcjSW1/jf1CHgwzXKBRwfkwCgYIKoZIzj0EAwQwFTET
...
...
u93naKM2Su+gyBo=
-----END CERTIFICATE-----
```

6. Save the `va.crt` file so you can transfer it during the Entrust Validation Authority GUI Configuration steps of the Certificate Authority.

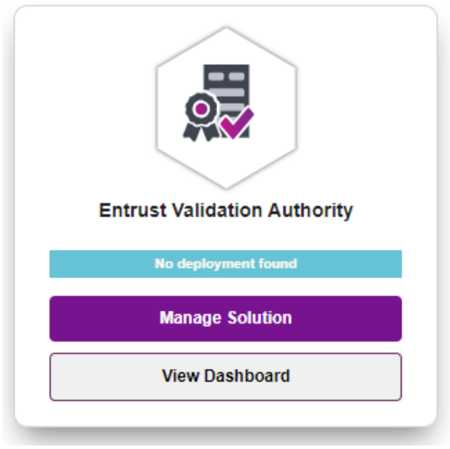
2.7. Configure Entrust Validation Authority from the web UI

1. Sign in to the Entrust Deployment Management Console at the following URL:

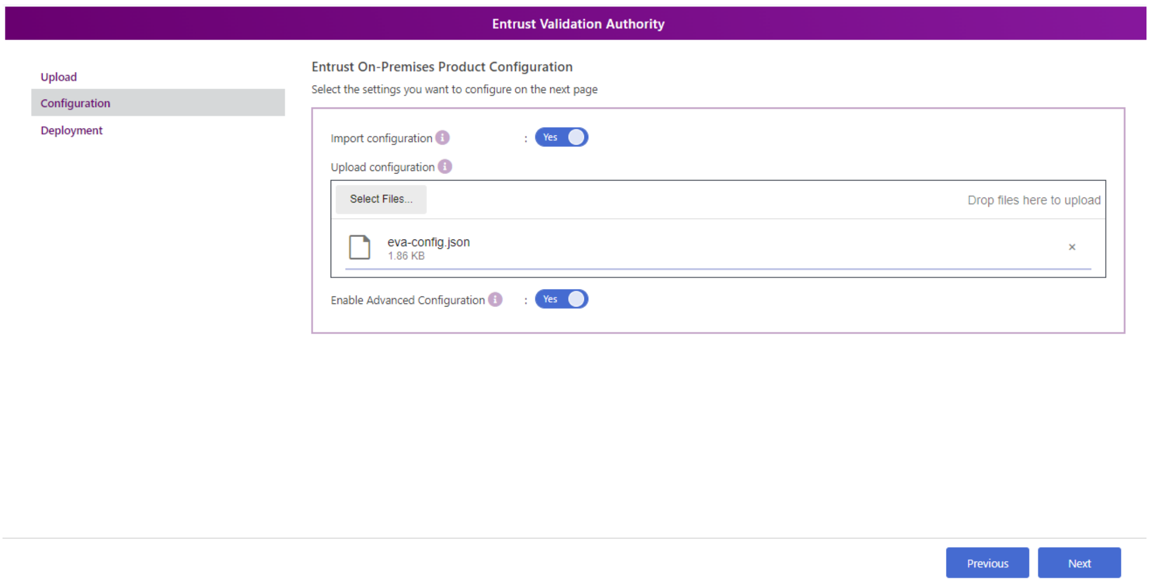
   ```
   https://<edm_host>/management-console
   ```

   Use the **admin** account and password that you configured during setup.

2. In the content pane, select **Manage Solution** for **Entrust Validation Authority**.



3. Select **Configuration** on the left.
4. In the **Entrust On-Premises Product Configuration** page, activate the **Import configuration** toggle switch, then select **Select Files** to import the sample configuration file included with the distribution package for Entrust Validation Authority.
5. Activate the **Enable Advanced** toggle to display advanced parameters on the next page. (Optional)


Entrust Validation Authority

Entrust On-Premises Product Configuration

Select the settings you want to configure on the next page

Import configuration : Yes

Upload configuration

Select Files... Drop files here to upload

eva-config.json 1.86 KB

Enable Advanced Configuration : Yes

6. Select **Next** to display the configuration options.
7. Configure the settings accordingly, then select **Next**.

Database name	Parameters for the connection to the database. In our example: **template1**.
Driver	The database driver name. In our example: **postgres**.
Max connections	The number of maximum concurrent database connections. Both the Status Feeder and the OCSP Responder use this value. Therefore, the database must support the double of that. In our example: **100**.
Host	The IP or hostname of the database host (**<db_host>**).
OCSP Responder User	OCSP Responder username (`ocspresponderuser`).
OCSP Responder password	Use the password, as configured earlier the database setup.
Status Feeder User	The Status Feeder username (`statusfeederuser`).
Status Feeder password	Use the password, as configured earlier the database setup.
Port	Leave the default port as **5432**.

SSL mode	Whether the EVA will use SSL in the DB connection or not. In our example: the postgres database is set to **disable**.

8. Configure the HSM settings.

Vendor	The vendor name of the HSM that will be used. For software cryptography set it to `none`. In our example: **nshield**.
Token label	The label of the HSM token that contains the private keys. It depends on the type of HSM protection selected. In our example: we created a softcard to hold the keys. Set it to the softcard name **testSC**.
HSM PIN	The pin for the HSM (the passphrase used for the softcard). Use the password as configured when you created the softcard. If you are using module protection, the HSM PIN can be anything.

9. Select **Next**.
10. Under **OCS Responder-Server**, take the default settings and select **Next**.

    This section is only visible if the **Advanced Configuration** Toggle is enabled.

11. **LDAP Servers**:

    We will not use LDAP so remove all the server settings from the template.

12. Configure the certificate authorities

    This integration uses one certificate authority in the integration, so remove the two additional certificate authorities from the template:

    - Under **Certificate Authority 1**:

CA ID	The identifier of the CA that issues the certificates. If using CAGW, the ID must match the one in CAGW. If you are using CRL as the source, it can be any name. In our example: we used the name of the CA, **easyrsa**.
Certificates Source	The source of the certificates for this CA. For this integration, we used **CRL**.

    - Under **Certificate Revocation List**:

Wait to pull certs duration	How often will EVA check for new certificate events to update the DB. Set it to **30s**.

CRL warning time	The period during which to enable the expiration warning for the last processed CRL. Set it to 4h.
CRL Host Server	Select **HTTP**.

- **Under Certificate Revocation List in HTTP server**:

CRL HTTP URL	The URL of the HTTP server where the CRL is hosted. In our example: we deployed an Apache server and made the `crl.der` file available in the server. The URL: `http://<apache_host>/crl/crl.der`
Use SN Lists	Set it to **false**.

- Under **OCS Responder**:

Profile ID	The identifier of the profile for processing the certificate status before generating an OCSP response Set it to **CRLProfile** or **CRLProfileWithArchiveCutOff**. In our example: **CRLProfile**.
CA Certificate	The certificate as a `pem` file. It is the CA that issues the certificates for which EVA will give OCSP service. Upload the `ca.crt` file from `easy-rsa` in this case because this is the CA we are using.
VA Certificate	The certificate as a `pem` file. It is the VA that will be used to sign the OCSP responses. This is the va certificate we signed using `easy-rsa` (`va.crt` file).

13. Select **Next**.

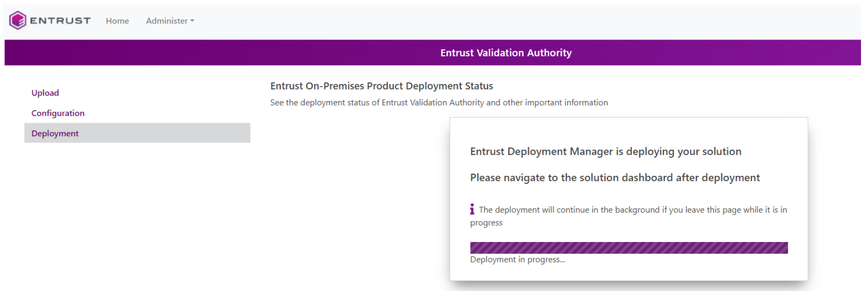
2.8. Submit the configuration settings

1. Select **Download** to download the new configuration.
2. Select **Validate** to validate the configured settings.
3. Correct any detected configuration error.
4. Select **Submit** and wait while Entrust Deployment Manager uploads the configuration and any attached files.

2.9. Entrust Validation Authority Deployment

After you **Submit** the Entrust Validation Authority Configuration and there are no errors reported in the configuration, the system is ready to be deployed:

1. Select **Deploy**.
2. Select **Yes** in the confirmation dialog.
3. Wait until the solution is deployed.



Entrust On-Premises Product Deployment Status

See the deployment status of Entrust Validation Authority and other important information

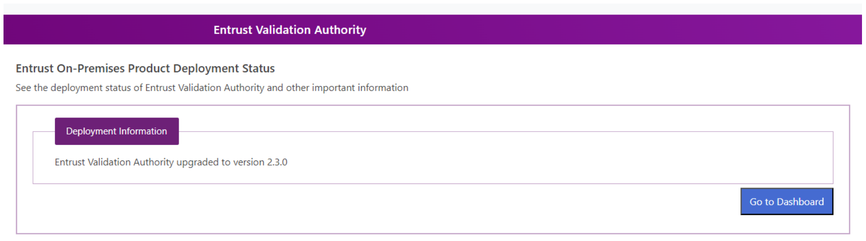
Entrust Deployment Manager is deploying your solution

Please navigate to the solution dashboard after deployment

The deployment will continue in the background if you leave this page while it is in progress

Deployment in progress...

4. Once deployed, select **Go to Dashboard**.

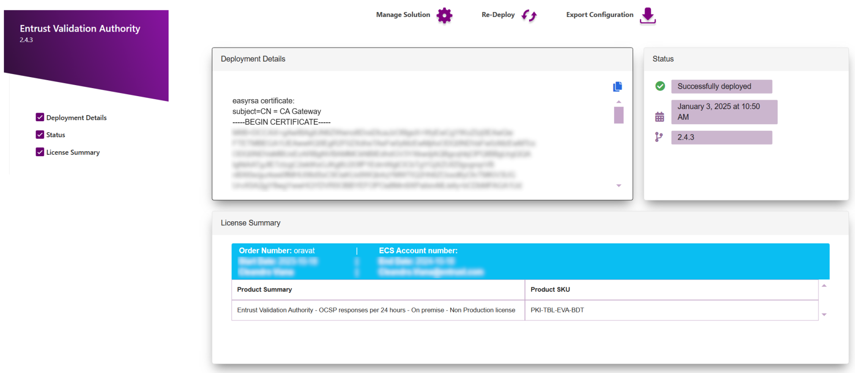


Entrust On-Premises Product Deployment Status

See the deployment status of Entrust Validation Authority and other important information

Entrust Validation Authority upgraded to version 2.3.0

5. You should see that EVA has been deployed successfully.



Deployment Details

```
easyrsa certificate:
subject=CN = CA Gateway
-----BEGIN CERTIFICATE-----
[redacted]
```

Status

Successfully deployed

January 3, 2025 at 10:50 AM

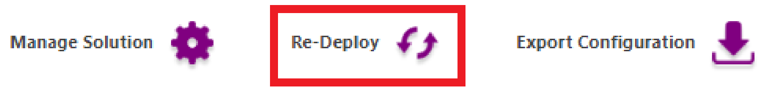
2.4.3

License Summary

Order Number: oravat | ECS Account number:
[redacted]

Product Summary	Product SKU
Entrust Validation Authority - OCSP responses per 24 hours - On premise - Non Production license	PKI-TBL-EVA-BDT

If the deployment fails, follow the instructions in the *Entrust Validation Authority Deployment Guide* to see how to check the logs. Sometimes if fails due to a timeout issue where the kubernetes pods do not come up in the allotted time frame, however the system still deploys sucessfully. To fix the failure in the dashboard, just redeploy EVA using the **Re-Deploy** function in the UI.



To check that the system is up, run the following command in the EDM server:

```
% sudo kubectl get pods -n eva

NAME                                 READY   STATUS    RESTARTS    AGE
eva-ocspresponder-7c9d5fcc79-w69z7   2/2     Running   0           6m10s
eva-ocspresponder-7c9d5fcc79-zfwjr   2/2     Running   0           6m10s
eva-statusfeeder-655d6c98fd-dv5d4    1/1     Running   0           6m9s
eva-crlshim-0-0                      1/1     Running   0           5m37s
```

You can also run the EVA testing on the next section. It should be successful if the pods above are running.

2.10. Entrust Validation Authority Testing

After deploying Entrust Validation Authority, you can test the OCSP Responder service as follows.

- With OpenSSL
- With the health check endpoint

2.10.1. Test the OCSP Responder with OpenSSL

Run the following `openssl` command to test the OCSP Responder service.

```
% openssl ocsp -issuer **ca_cert** -serial **sn** -url **url** -VAfile **va_cert**
```

If the serial number is not found in the CRL, EVA will return **good** status. If the serial is in the CRL, it'll return the revoke information contained in the CRL

2.10.1.1. Test the OCSP Responder with a serial number that is not in the CRL

```
% openssl ocsp -issuer ./ca.crt -serial 0x000000002439fa8f5fe6370bb20ccb2556da6991 -url http://<host>/eva -VAfile
./va.crt

Response verify OK
0x000000002439fa8f5fe6370bb20ccb2556da6991: good
        This Update: Feb  3 20:41:08 2026 GMT
        Next Update: Aug  2 20:41:08 2026 GMT
```

2.10.1.2. Test the OCSP Responder with a serial number that is in the CRL

```
% openssl ocsp -issuer ./ca.crt -serial 0xE2B70CDE937377589FEBFF849A25636E -url http://<host>/eva -VAfile
./va.crt

Response verify OK
0xE2B70CDE937377589FEBFF849A25636E: revoked
        This Update: Feb  3 20:41:08 2026 GMT
        Next Update: Aug  2 20:41:08 2026 GMT
        Revocation Time: Feb  3 20:40:11 2026
```

The serial number can be found in the output for the command `./easyrsa revoke cagw`, which you ran in an earlier step.

2.10.2. Test the OCSP Responder with the health check endpoint

Entrust Validation Authority exposes the following endpoint to check the health of the database and HSM connections.

```
http://<edm-host>/eva/health
```

This endpoint returns a HTTP 503 response when the health check fails.

```
$ wget --debug http://<edm-host>/eva/health


--- Request
GET /eva/health HTTP/1.1
Host: x.x.x.x
User-Agent: toybox wget/0.8.10
Connection: close

--- Response
HTTP/1.1 200 OK
date: Wed, 04 Feb 2026 17:45:28 GMT
content-length: 0
x-envoy-upstream-service-time: 10
x-content-type-options: nosniff
server: istio-envoy
connection: close
```

2.11. FIPS Level 3 remarks and recommendations

Recommendations when a FIPS Level 3 world file is used for the HSM configuration:

- Create an OCS card 1/N where N is the number of HSMs in the configuration.
- All HSMs in the configuration must use the same world file.
- Leave the OCS card inserted on each HSM used in the configuration.

- The OCS card is only used for FIPS authorization and not to protect the keys.
- The OCS card must be present any time new key material is created.

Chapter 3. Additional resources and related products

3.1. nShield HSMs

3.2. nShield as a Service

3.3. Entrust products

3.4. nShield product documentation