Red Hat Certificate System

nShield® HSM Integration Guide

2026-11-02

© Entrust Corporation. All rights reserved.

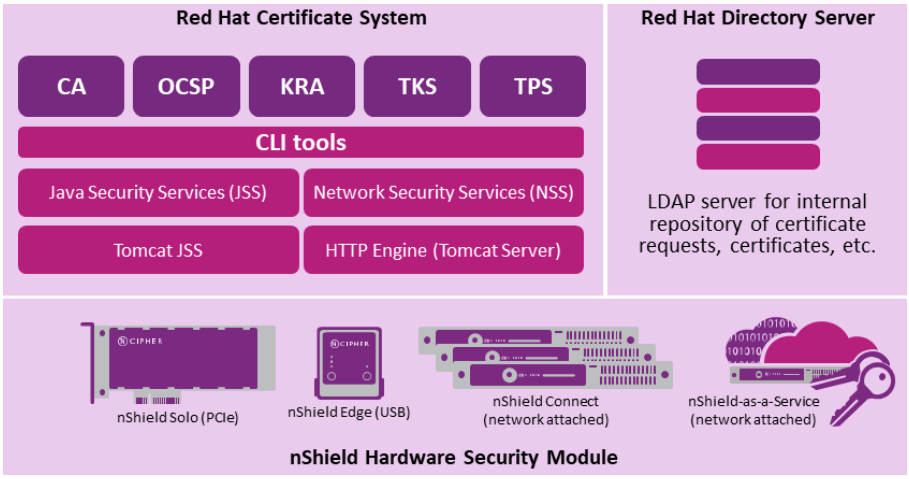
Table of Contents

1. Introduction . . . 1
    1.1. Requirements . . . 1
    1.2. Licensing . . . 2
    1.3. Product configurations . . . 2
    1.4. Supported nShield functionality . . . 3
    1.5. Policy requirements . . . 4
2. Procedures . . . 6
    2.1. Install the Operating System . . . 6
    2.2. Configure the HSM . . . 9
    2.3. Install RHDS . . . 12
    2.4. Install RHCS . . . 15
    2.5. Import the CA chain and user credentials into Firefox . . . 25
    2.6. Basic system tests . . . 26
3. Additional Testing . . . 28
    3.1. Before you begin . . . 28
    3.2. Test 1: The HSM and Security World are usable . . . 32
    3.3. Test 2: The CA service is running and serving HTTPS . . . 34
    3.4. Test 3: The CA system keys are stored on the HSM token . . . 35
    3.5. Test 4: Agent login with the admin certificate . . . 37
    3.6. Test 5: Issue a certificate signed by the HSM-protected CA key . . . 39
    3.7. Test 6: OCSP reports the new certificate as good . . . 42
    3.8. Test 7: Revoke the certificate . . . 43
    3.9. Test 8: OCSP reports the certificate as revoked . . . 44
    3.10. Test 9: The certificate record is stored in RHDS . . . 44
    3.11. Test 10: Generate and verify a new CRL . . . 46
    3.12. Test 11: Verify the signed audit log . . . 48
    3.13. Test 12: CA restart and unattended HSM token login . . . 50
    3.14. Test 13: The CA cannot sign while the HSM is unavailable (expected failure) . . . 51
    3.15. Clean up . . . 53
4. Configure pkispawn . . . 54
    4.1. Modifying the sample pkispawn configuration file . . . 54
5. Additional resources and related products . . . 59
    5.1. nShield HSMs . . . 59
    5.2. nShield as a Service . . . 59
    5.3. Entrust products . . . 59
    5.4. nShield product documentation . . . 59

Chapter 1. Introduction

This guide explains how to configure a Red Hat Certificate System (RHCS) installation with an Entrust nShield Hardware Security Module (HSM). The integration between the HSM and Red Hat Certificate System uses the PKCS #11 cryptographic API.

The following diagram shows the basic architecture of an RHCS deployment:



This guide does not cover every step in the process of setting up all software. Some packages require that other packages are already configured, initialized, and running before they can be installed successfully.

1.1. Requirements

For an RHCS installation, you need to set up a Red Hat Enterprise Linux system. In principle, a CentOS platform works the same way. However, its core RHCS packages might not be as up to date as those provided by Red Hat.

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

Requirements for the Red Hat Enterprise Linux server:

Component	Minimum Requirements	Recommended Requirements
Memory	2 GB	4 GB or more
Processor	1 CPU	1 CPU or more
Processor Cores	2	4 or more, AES-NI support

Component	Minimum Requirements	Recommended Requirements
Hard Disk	20 GB	80 GB or more
CD/DVD	Optional	Optional
Network Adapter	1	1
USB Controller	Only required for nShield Remote Administration	
Display	Standard configuration	

Components required for installation:

- Security World software v13.6.18.

If you use post-quantum algorithms, use Security World Software v13.9.6.

- Red Hat Enterprise Linux v10.2.
- Linux firewall (`firewalld`).
- Static IP address.
- Mozilla Firefox.
- OpenJDK 64-bit.
- Red Hat Directory Server (RHDS) v13.2

  Any LDAP-compliant database should be compatible with the integration.

- Red Hat Certificate System (RHCS) v11.10.1

1.2. Licensing

You do not need to import a license into the product after installation. Contact Red Hat for appropriate licensing to purchase RHCS product support and RHN channel access.

1.3. Product configurations

RHCS v11.10.1 has been tested with the following nShield HSM configurations:

nShield HSM	Software	Firmware	Netimage	FIPS-140 Level 3	Ciphersuite
Connect XC	13.6.18	12.72.4 (FIPS 140-2 certified)	13.6.18	Yes	DLf3072s256mA EScSP800131Ar1
nShield 5C	13.6.18	13.4.5 (FIPS 140-3 certified)	13.6.18	Yes	DLf3072s256mA EScSP800131Ar1

The Post Quantum edition has been tested with the following settings:

nShield HSM	Software	Firmware	Netimage	FIPS-140 Level 3	Ciphersuite
Connect XC	13.9.6	13.8.5	13.9.6	No	ECp521mAES
nShield 5C	13.9.6	13.8.6	13.9.6	No	ECp521mAES

Security World Software v13.9.6 supports post-quantum algorithms and was used to test post-quantum functionality.

The Common Criteria edition has been tested with the following settings:

nShield HSM	Software	Firmware	Netimage	Security World	Ciphersuite
Connect XC	13.6.18	12.60.15	13.6.15	Common Criteria	DLf3072s256mA EScSP800131Ar1
nShield 5C	13.6.18	13.5.1	13.6.15	Common Criteria	DLf3072s256mA EScSP800131Ar1

1.4. Supported nShield functionality

Red Hat Certificate System does not support module-protected keys. When you enable the use of an HSM, RHCS requires a token name, but module-protected keys do not have one. Using "accelerator" as the token name does not work.

Feature	Support	Feature	Support
Key Generation	Yes	Module-only keys	No
Key Management	Yes	FIPS 140 Level 3 mode support	Yes
1-of-N Operator Card Set	Yes	Common Criteria mode support	Yes
K-of-N Operator Card Set	Yes	Load Sharing	Yes
Softcards	Yes	Failover	Yes
Post Quantum	Yes	ML-DSA Algorithms	Yes

1.5. Policy requirements

Entrust recommends that your organization operates its PKI using an approved organizational Certificate Policy, Certificate Practices Statement, and any other policy/procedure guidance necessary to govern the administration of the PKI and associated HSM(s). In particular, these documents should specify the following aspects of HSM administration:

- The number and quorum of Administrator Cards in the Administrator Card Set (ACS), and the policy for managing these cards.

  Common Criteria firmware requires a minimum quorum of two Administrator Cards.

- Whether application keys are to be protected by Operator Card Set (OCS) or Softcard protection mechanisms. The module key protection mechanism is not supported.
- The number and quorum of Operator Cards in the OCS (if OCS key protection is used), and the policy for managing these cards.
- Whether the Security World should be compliant with FIPS 140 Level 3, FIPS 140 Level 2, or Common Criteria.

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

  See the *User Guide* for your HSM and the *nShield Security Manual* for more recommendations, or contact your local Entrust nShield Account Manager to arrange a technical discussion on architecture and best practices specific to your environment.

Compliance with US government security standards

It may be important for your organization to configure your RHEL OS in compliance with applicable standards such as FIPS.

- For Red Hat product compliance with government and industry security standards, see https://access.redhat.com/compliance.
- For information on enabling FIPS mode for RHEL v10, see https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html/security_hardening/switching-rhel-to-fips-mode

Chapter 2. Procedures

The instructions in this guide refer to one server, with two network interfaces and unique IP addresses, on which the Red Hat applications are installed:

Application	Domain	IP Address
Red Hat Certificate System (RHCS)	pki.domain.com	10.0.0.2
Red Hat Directory Server (RHDS)	ldap.domain.com	10.0.0.3

When you follow the instructions, use the domain names and IP addresses of your system.

2.1. Install the Operating System

1. Install Red Hat Enterprise Linux server on your target server platform or virtualized environment.

   No GUI is required for the installation of RHCS. However, to perform agent functions on the CA, you must use a system with a web browser (Mozilla Firefox is preferred).

2. If FIPS mode is required, ensure that the system is operating in FIPS mode.

   To enable the cryptographic module self-checks mandated by the Federal Information Processing Standard (FIPS) 140-3, you must operate RHEL 10 in FIPS mode. The only correct way to switch the system to FIPS mode is to enable it during the RHEL installation. For more information, see Switching RHEL to FIPS mode.

3. Ensure that the system is operating in SELinux enforcing mode:

   ```
   % getenforce

   Enforcing
   ```

   If the result is Permissive or Disabled, you need to enable SELinux before continuing.

   If the status was Permissive:

   a. Make SELinux enforcing on reboot by editing `/etc/selinux/config`. Change `SELINUX=permissive` to `SELINUX=enforcing`.

   b. Set SELinux to enforcing for this session:

      ```
      % setenforce 1
      ```

c. Check SELinux status again:

```
% getenforce
```

It should now show Enforcing.

If the status was Disabled:

a. Make SELinux enforcing on reboot by editing `/etc/selinux/config`. Change `SELINUX=disabled` to `SELINUX=enforcing`.

b. Enable a `relabel` operation on reboot:

```
touch /.autorelabel
```

c. Reboot the server.

d. Check SELinux status again:

```
getenforce
```

It should now show Enforcing.

4. Ensure that the `firewalld` service is enabled and running:

```
systemctl enable firewalld
systemctl start firewalld
```

5. Configure the system time zone.
For example, to use UTC:

```
timedatectl set-timezone UTC
timedatectl set-time "YYYY-MM-DD HH:MM:SS"
(HH:MM:SS is in 24-hour time)
```

If you are using NTP, see the Red Hat Enterprise Linux documentation.

6. Configure the system hostname:

```
hostnamectl set-hostname <pki.domain.com>
```

Where *<pki.domain.com>* is the FQDN of your server.

7. If you are using the nShield Connect model, configure network interfaces as needed for external access to the RHCS subsystems and for the HSM.

If you are installing the RHDS component on an external system, a network interface must be able to communicate with that system as well.
For instructions on configuring the network interfaces for your system, for example by using `nmcli`, see the Red Hat Enterprise Linux documentation.

For example:

a. Modify `ens33` so that its IP address is 10.0.0.2:

```
nmcli connection modify ens33 connection.autoconnect yes \
  ipv4.method manual ipv6.method auto \
  ipv4.addresses 10.0.0.2/24 ipv4.gateway 10.0.0.1 ipv4.dns 10.0.0.1
```

b. Add a second IP address, 10.0.0.3, to `ens33`:

```
nmcli connection modify ens33 connection.autoconnect yes \
  ipv4.method manual ipv6.method auto \
  +ipv4.addresses 10.0.0.3/24 +ipv4.gateway 10.0.0.1 +ipv4.dns 10.0.0.1
nmcli connection up ens33
```

8. If you are not using DNS, configure `/etc/hosts` to include any aliases that might be used for the RHCS and RHDS subsystems.

   If DNS or `/etc/hosts` is not configured to resolve the applicable hostnames, services, especially RHDS, might hang for a long time when they are started.

   The following is an example configuration, assuming the CA and LDAP are on separate servers:

```
127.0.0.1 localhost
10.0.0.2 pki.domain.com pki
10.0.0.3 ldap.domain.com ldap
```

9. Register the system:

```
sudo subscription-manager register
```

10. Configure the `ulimits` for the OS by adding the following lines to the end of the `/etc/security/limits.conf` file:

```
root soft nofile 65536
root hard nofile 65536
```

11. Configure any system security settings and lockdown procedures, such as screensaver settings, to ensure that the system is secure before generating the CA private key(s).

2.2. Configure the HSM

Install the HSM by following the instructions in the *Installation Guide* for the HSM.

2.2.1. Install the nShield Security World software

To install the Security World software and create the Security World:

1. Uninstall any existing Security World software before installing the new Security World software.
2. Install the latest version of the Security World software as described in the *User Guide* for the HSM.

2.2.2. Enroll the new HSM client with the HSM

1. Enroll the client:
   - For a privileged client, such as the RFS, if enrolling as a client:

     ```
     nethsmenroll --privileged HSM_IP_ADDRESS
     ```

   - For an unprivileged client, which includes most application servers, such as a CA or an OCSP responder:

     ```
     nethsmenroll HSM_IP_ADDRESS
     ```

2. Verify enrollment with the `enquiry` command:

   ```
   enquiry -m 1
   ```

2.2.3. Configure cknfastrc for RHCS

1. Configure the `/opt/nfast/cknfastrc` file with the following settings:

Key Protection Mechanism	CKNFASTRC Configuration Parameters
Module protection	N/A: RHCS does not support this option

Key Protection Mechanism	CKNFASTRC Configuration Parameters

Softcard protection	`CKNFAST_OVERRIDE_SECURITY_ASSURANCES=wrapping_crypt`
	`CKNFAST_LOADSHARING=1`
	`CKNFAST_NO_ACCELERATOR_SLOTS=1`

OCS protection
(K-of-N with k=1)
	`CKNFAST_OVERRIDE_SECURITY_ASSURANCES=wrapping_crypt`
	`CKNFAST_LOADSHARING=1`
	`CKNFAST_NO_ACCELERATOR_SLOTS=1`

OCS protection
(K-of-N with k > 1)
	`CKNFAST_OVERRIDE_SECURITY_ASSURANCES=wrapping_crypt`
	`CKNFAST_LOADSHARING=1`
	`CKNFAST_NO_ACCELERATOR_SLOTS=1`
	`NFAST_NFKM_TOKENSFILE=/opt/nfast/nfast-nfkm-tokensfile`

2. To troubleshoot the CA installation, check the following logs:
   - RHCS logs, in `/var/log/pki/<instance>`.
   - nShield hardserver logs, in `/opt/nfast/log/hardserver.log`.

2.2.4. Create or load a Security World

1. Initialize a Security World as described in the *User Guide* for the HSM.

   The decision to create a new Security World or to load an existing Security World is based on your organization's security practices.

2.2.5. Create an OCS or Softcard to protect the application keys

If you already have an OCS or a Softcard to use from an existing Security World:

1. Make sure that the appropriate card or Softcard files are in `/opt/nfast/kmdata/local`.
2. Continue with the instructions in Install RHDS.

2.2.5.1. Method #1: Create an OCS

This `createocs` command is an example. Use parameters that satisfy your organization's security requirements.

1. Use the following parameters to create a new OCS:
    - K-of-N defined as 1/3.
    - No time-out.
    - Persistent card set.
    - Do not name cards individually.
    - Enable PIN recovery.
    - Pick a name for your OCS. The example below uses OCS1.

```
createocs -m 1 -N "OCS1" -Q 1/3 -T 0 --persist --pp-recovery
```

2. If you are using a FIPS 140 Level 3 Security World, you need to present an ACS or OCS card for FIPS authorization.
3. Enter a password for each OCS card in the quorum. Each card can have a unique password, but this is not recommended for most use cases.
4. Verify that the OCS card set is visible:

```
nfkminfo -c
```

2.2.5.2. Method #2: Create a Softcard

This `ppmk` command is an example. Use parameters that satisfy your organization's security requirements.

1. Create a new Softcard:
    - If you are using a FIPS 140 Level 3 Security World, you need to present an ACS or OCS to provide FIPS authentication.
2. Pick a name for your Softcard. The example below uses `SOFTCARD1`.
3. Make the Softcard container recoverable.

```
ppmk --new --recoverable SOFTCARD1
```

4. Verify that the Softcard was created:

```
ppmk --list
```

2.2.5.3. Copy KMDATA files to other HSM client systems

The Security World files are located in the `/opt/nfast/kmdata/local` folder on the system where they were generated.

To use the Security World on other HSM clients, copy the following KMDATA files from `/opt/nfast/kmdata/local` to your other HSM clients:

- `world`
- `cards_*`
- `card_*`
- `softcard_*`
- `module_*`

You can copy the files manually, using tools such as `rsync` or `scp`, or by configuring the `rfs-sync` capability of the nShield tools.

For information on configuring and using `rfs-sync`, see the *User Guide* for your HSM.

2.3. Install RHDS

You need the Red Hat Directory Server packages from the appropriate Red Hat Network channel. Some packages are in the RHEL OS channel, and some are in the RHDS channel. For instructions to install these packages, see the Red Hat Directory Server documentation.

These steps use the instance name ca-1. Entrust recommends that you use a more descriptive identifier to suit the requirements of your organization. It is most convenient to name the RHDS instance the same as the RHCS instance, especially if you plan to create multiple instances of either on their respective servers.

For instructions to enable LDAPS on the directory server before installing RHCS, see the Red Hat Directory Server documentation. There are generally two methods for this:

- Existing CA:
  Create a TLS key/request and have the certificate signed by an already deployed CA, if one exists.
- New CA:
  a. Create a TLS key/request.
  b. Self-sign it for temporary use in the LDAP server until the new RHCS CA is deployed.
  c. Re-sign the TLS certificate request on the new RHCS CA.

d. Remove the self-signed certificate from the LDAP server's NSS database.

e. Install the new TLS certificate along with the new CA chain.

In the following steps, the RHDS instance is installed on the same system as RHCS. However, you can install them on separate servers.

1. Enable the default RHEL repositories:

```
sudo subscription-manager repos \
    --enable=rhel-10-for-x86_64-appstream-rpms \
    --enable=rhel-10-for-x86_64-baseos-rpms
```

2. Enable the default Directory Server repository:

```
sudo subscription-manager repos --enable=dirsrv-13-for-rhel-10-x86_64-rpms
```

3. Install the RHDS packages and dependencies:

```
sudo dnf install redhat-ds
```

4. Check the installation:

```
sudo subscription-manager list

+-----------------------------------------+
    Installed Product Status
+-----------------------------------------+
Product Name: Red Hat Directory Server
Product ID:   200
Version:      13.2
Arch:         x86_64

Product Name: Red Hat Enterprise Linux for x86_64
Product ID:   479
Version:      10.2
Arch:         x86_64
```

5. Open the ports in the appropriate firewall zone:

```
firewall-cmd --permanent --add-port={389/tcp,636/tcp} --zone=<zone>
firewall-cmd --reload
```

6. Configure the LDAP service account group:

```
groupadd -r ldap
```

7. Configure the LDAP service account user:

```
useradd -g ldap ldap-ca-1
```

8. Configure RHDS:

```
% dscreate interactive

# Here is an example:

Enter system's hostname [pki.domain.com]: ldap.domain.com

Enter the instance name [ldap]: ca-1

Enter port number [389]:

Create self-signed certificate database [yes]:

Enter secure port number [636]:

Enter Directory Manager DN [cn=Directory Manager]:

Enter the Directory Manager password:
Confirm the Directory Manager Password:

Choose whether mdb or bdb is used. [mdb]:

Enter the lmdb database size [20.0 GB]:

Enter the database suffix (or enter "none" to skip) [dc=ldap,dc=domain,dc=com]: o=ca-1-CA

Create sample entries in the suffix [no]:

Create just the top suffix entry [no]:

Do you want to start the instance after the installation? [yes]:

Are you ready to install? [no]: yes
Starting installation ...
Validate installation settings ...
Create file system structures ...
Create self-signed certificate database ...
Perform SELinux labeling ...
Create database backend: o=ca-1-CA ...
Perform post-installation tasks ...
Completed installation for instance: slapd-ca-1
```

You can control the RHDS instance with the following commands.

Function	Command
Enable on boot	`systemctl enable dirsrv@<RHCS-instance>`
Start manually	`systemctl start dirsrv@<RHCS-instance>`
Stop manually	`systemctl stop dirsrv@<RHCS-instance>`
Restart manually	`systemctl restart dirsrv@<RHCS-instance>`

9. Test the RHDS connection:

```
% sudo ldapsearch -o ldif-wrap=no -x -LLL \
  -H ldap://ldap.domain.com:389 \
  -D 'cn=Directory Manager' -W \
  -b 'cn=config' -s base '(objectclass=*)' \
  nsslapd-versionstring nsslapd-defaultnamingcontext

Enter LDAP Password:
dn: cn=config
nsslapd-versionstring: 389-Directory/3.2.0
nsslapd-defaultnamingcontext: o=ca-1-ca
```

2.4. Install RHCS

You need the Red Hat Certificate System packages from the appropriate Red Hat Network channel. The CA and KRA subsystems are part of the RHEL OS channel, but the TKS and TPS subsystems require access to the Red Hat Certificate System channel. Many dependency packages must be installed along with the RHCS packages, including `apache/httpd` and `apache/tomcat`. For package installation instructions, see the Red Hat Certificate System documentation.

The various subsystems for Red Hat Certificate System are installed and configured individually. The initial installation is performed using package management tools such as `dnf`. Each subsystem is set up with the `pkispawn` command-line tool.

2.4.1. Install the packages

The `redhat-pki` package installs packages to support all RHCS subsystems.

1. Enable the repository:

```
sudo subscription-manager repos --enable=certsys-11.0-for-rhel-10-x86_64-rpms
```

2. Install the Certificate System packages:

```
sudo dnf install redhat-pki
```

3. Check the installation:

```
sudo subscription-manager list

+-------------------------------------------+
    Installed Product Status
+-------------------------------------------+
Product Name: Red Hat Directory Server
```

```
Product ID:   200
Version:      13.2
Arch:         x86_64

Product Name: Red Hat Certificate System
Product ID:   216
Version:      11.0
Arch:         x86_64

Product Name: Red Hat Enterprise Linux for x86_64
Product ID:   479
Version:      10.2
Arch:         x86_64
```

4. Restore the SELinux context on `/opt/nfast` to account for the new `pki-selinux` policy:

```
sudo restorecon -FRvv /opt/nfast
```

Ensure that your RHEL OS packages are updated, in particular `httpd`, `tomcat`, and `nss`. RHCS packages do not always have the correct dependency versions, and sometimes key generation might fail because `nss` needs to be updated to the latest version.

5. Restart the nfast hardserver:

```
sudo /opt/nfast/sbin/init.d-ncipher restart
```

2.4.2. Check the PKCS #11 connection to the HSM

2.4.2.1. Check with Mozilla NSS database tools

1. Create the temporary NSS database:

```
sudo mkdir -p /opt/tempnssdb
cd /opt/tempnssdb
sudo modutil -dbdir . -create -force
```

2. Add the nShield PKCS #11 library to the temporary NSS database:

```
modutil -dbdir . -add nfast \
  -libfile /opt/nfast/toolkits/pkcs11/libcknfast.so -force
Module "nfast" added to database.
```

3. After the NSS database is created and linked to the HSM, check the PKCS #11 token information with the RHCS tools:

```
% sudo TokenInfo .
```

```
Database Path: .
Found external module 'NSS Internal PKCS #11 Module'
Found external module 'nfast'
Found external module 'p11-kit-proxy'
Found external token 'testSC'
```

4. Remove the temporary NSS database:

```
sudo rm -rf /opt/tempnssdb
```

2.4.2.2. Check using HSM tools

Use the `ckcheckinst` command to test the PKCS #11 installation with nShield tools. This example uses an OCS. Modify it if you are using Softcard protection.

In some cases, `ckcheckinst` might fail. However, this does not necessarily indicate a system configuration problem. Consult Entrust nShield Support if necessary.

```
sudo /opt/nfast/bin/ckcheckinst

PKCS#11 library interface version 3.2
                          flags 0
                  manufacturerID "nCipher Corp. Ltd              "
              libraryDescription "nCipher PKCS#11 13.9.6-628-875bd"
           implementation version 13.09
         Loadsharing and Failover enabled

Slot  Status           Label
====  ======           =====
   0  Operator card    "testOCS2                       "

Select slot number to run library test or 'R'etry or to 'E'xit: 0
Using slot number 0.

Please enter the passphrase for this token (No echo set).
Passphrase:

Test                     Pass/Failed
----                     -----------

1 Generate RSA key pair  Pass
2 Generate DSA key pair  Pass
3 Encryption/Decryption  Pass
4 Signing/Verification   Pass

Deleting test keys        ok

PKCS#11 library test successful.
```

2.4.3. Configure the firewall

1. Open ports to support RHCS functions. These ports are the default for RHCS. If you plan to use non-default ports, add those ports to the firewall instead of the ones listed in this table.

Service	Port
CA HTTP Proxy	8080/tcp
CA HTTPS Proxy	8443/tcp
CA Security Domain	8443/tcp
CA Tomcat Server	8005/tcp
CA AJP	8009/tcp

2. Configure the firewall:

```
sudo firewall-cmd --permanent --add-port={8080/tcp,8443/tcp,8005/tcp,8009/tcp}
sudo firewall-cmd --reload
```

2.4.4. Configure service account users and groups

Create the appropriate user and group accounts before staging the system.

Members of the `pkiadmin` system group have full access to tasks in the agent service interface:

```
groupadd -r pkiadmin
```

Members of the `pkiaudit` system group can read the signed audit logs.

```
groupadd -r pkiaudit
```

To create a new service account user and assign it to the `nfast`, `pkiadmin`, and `pkiaudit` groups:

```
useradd -g pkiuser -G nfast,pkiadmin,pkiaudit \
  -d /usr/share/pki -s /sbin/nologin \
  -c "RHCS ca-1" -r pkiuser-ca-1
```

2.4.5. Pre-configure the RHCS instance

Run `pkispawn` for an initial creation phase so you can pre-configure parameters, such as certificate distinguished names and validity periods. For information on the two-step installation, see the *RHCS Installation Guide*.

Do not modify `/usr/share/pki/server/etc/default.cfg` directly.

1. Create a copy of `/usr/share/pki/server/etc/default.cfg`, for example, at `/opt/pkispawn.cfg`.
2. Modify it to match your system.

   For information on how to modify the default configuration file, see:

   - Configure pkispawn.
   - `man 5 pki_default.cfg`.
   - *Red Hat Certificate System Planning, Installation, and Deployment Guide*.

If you are using a FIPS Level 3 world file, have an OCS card connected to provide FIPS authorization. For post-quantum keys, use the PQ ML-DSA-XX algorithms in the configuration. Update the `/opt/nfast/cknfastrc` file according to the protection method that you are using.

1. Generate an INF for subsystem setup.

   If you are using OCS-protected keys and K>1, use the preload command with `pkispawn`:

   ```
   sudo preload -m <module number> -f "<preload FilePath>" \
     --cardset-name=<OCS Cardset-Name> \
     pkispawn -f /path/to/pkispawn.cfg -s CA --skip-configuration
   ```

   Insert the OCS cards and, if prompted, enter the OCS passphrase.

   For other OCS scenarios and other protection methods, run `pkispawn` on its own:

   ```
   sudo pkispawn -f /path/to/pkispawn.cfg -s CA --skip-configuration
   ```

2. At this point, you can modify any of the pre-configuration files as necessary. See the information on two-step installation in the *RHCS Installation Guide*.

The output of the `pkispawn` commands above should resemble the following:

```
Loading deployment configuration from /opt/pkispawn.cfg.
Installing CA into /var/lib/pki/ca-1.
Created symlink '/etc/systemd/system/pki-tomcatd.target.wants/pki-tomcatd@ca-1.service' →
'/usr/lib/systemd/system/pki-tomcatd@.service'.
```

```
    ==========================================================================
                              INSTALLATION SUMMARY
    ==========================================================================

      The CA subsystem of the 'ca-1' instance
      must still be configured!

      To check the status of the subsystem:
            systemctl status pki-tomcatd@ca-1.service

      To restart the subsystem:
            systemctl restart pki-tomcatd@ca-1.service

      The URL for the subsystem is:
            https://pki.domain.com:8443/ca

      PKI instances will be enabled upon system boot

    ==========================================================================
```

2.4.6. Configure the CA instance

Determine whether you are creating a root CA or a subordinate (issuing) CA, then define an appropriate configuration file for `pkispawn` by using one of the following procedures:

2.4.6.1. Create a root CA

Set `pki_external=False` for the `pkispawn.cfg` file so that `pkispawn` self-signs the `caSigningCert` object.

If you are using OCS-protected keys and K>1, use the preload command with `pkispawn`:

```
preload -m <module number> -f "<preload FilePath>" \
  --cardset-name=<OCS Cardset-Name> \
  pkispawn -s CA --debug -f /path/to/pkispawn.cfg --skip-installation
```

Insert the OCS cards and, if prompted, enter the OCS passphrase.

For other OCS scenarios and other protection methods, run the single-phase `pkispawn` on its own:

```
pkispawn -s CA --debug -f /path/to/pkispawn.cfg --skip-installation
```

The output should resemble the following:

```
...
...
    ==========================================================================
                              INSTALLATION SUMMARY
    ==========================================================================
```

```
    Administrator's username:             caadmin
    Administrator's PKCS #12 file:
          /root/.dogtag/ca-1/ca_admin_cert.p12

    To check the status of the subsystem:
          systemctl status pki-tomcatd@ca-1.service

    To restart the subsystem:
          systemctl restart pki-tomcatd@ca-1.service

    The URL for the subsystem is:
          https://pki.domain.com:8443/ca

    PKI instances will be enabled upon system boot

  ================================================================================
```

2.4.6.2. Check the status of the deployment

1. Check the status of the service:

```
sudo systemctl status pki-tomcatd@ca-1.service

     Loaded: loaded (/usr/lib/systemd/system/pki-tomcatd@.service; enabled; preset: disabled)
     Active: active (running) since Tue 2026-10-13 20:36:44 UTC; 4min 26s ago
 Invocation: 165b7b2f63644a30afdbbc309a536f3a
    Process: 42121 ExecStartPre=/usr/sbin/pki-server upgrade ca-1 (code=exited, status=0/SUCCESS)
    Process: 42123 ExecStartPre=/usr/sbin/pki-server migrate ca-1 (code=exited, status=0/SUCCESS)
    Process: 42176 ExecStartPre=/usr/bin/pkidaemon start ca-1 (code=exited, status=0/SUCCESS)
   Main PID: 42194 (java)
      Tasks: 95 (limit: 22820)
     Memory: 216.7M (peak: 218.4M)
        CPU: 17.428s
     CGroup: /system.slice/system-pki\x2dtomcatd.slice/pki-tomcatd@ca-1.service
             └─42194 /usr/lib/jvm/jre-21-openjdk/bin/java -Dcom.redhat.fips=false -Dredhat.crypto
-policies=false --add-opens=java.base/j>

Oct 13 20:36:50 pki.domain.com pki-tomcat-start[42194]: SLF4J: Found binding in
[jar:file:/usr/share/pki/server/webapps/pki/>
Oct 13 20:36:50 pki.domain.com pki-tomcat-start[42194]: SLF4J: Found binding in
[jar:file:/usr/share/pki/lib/slf4j-jdk14-1.7>
Oct 13 20:36:50 pki.domain.com pki-tomcat-start[42194]: SLF4J: See
http://www.slf4j.org/codes.html#multiple_bindings for an >
Oct 13 20:36:50 pki.domain.com pki-tomcat-start[42194]: SLF4J: Actual binding is of type
[org.slf4j.impl.JDK14LoggerFactory]
Oct 13 20:36:51 pki.domain.com pki-tomcat-start[42194]: SLF4J: Class path contains multiple SLF4J bindings.
Oct 13 20:36:51 pki.domain.com pki-tomcat-start[42194]: SLF4J: Found binding in
[jar:file:/usr/share/pki/ca/webapps/ca/WEB-I>
Oct 13 20:36:51 pki.domain.com pki-tomcat-start[42194]: SLF4J: Found binding in
[jar:file:/usr/share/pki/lib/slf4j-jdk14-1.7>
Oct 13 20:36:51 pki.domain.com pki-tomcat-start[42194]: SLF4J: See
http://www.slf4j.org/codes.html#multiple_bindings for an >
Oct 13 20:36:51 pki.domain.com pki-tomcat-start[42194]: SLF4J: Actual binding is of type
[org.slf4j.impl.JDK14LoggerFactory]
Oct 13 20:36:55 pki.domain.com pki-tomcat-start[42194]: Enter password for testOCS
```

2. Get the status over HTTPS:

```
curl -k https://pki.domain.com:8443/ca/admin/ca/getStatus
```

```
{
  "Response" : {
    "State" : "1",
    "Type" : "CA",
    "Status" : "running",
    "Version" : "11.10.1",
    "ProductVersion" : "Red Hat Certificate System 11.0.0"
  }
```

3. List the certificates:

```
sudo pki-server cert-find -i ca-1

  Cert ID: ca_signing
  Nickname: caSigningCert cert-ca-1 CA
  Token: testSC
  Serial Number: 0x87a66228bb8f66fb571cfe1d258f94e0
  Subject DN: CN=CA Signing Certificate,OU=ca-1,O=domain.com Security Domain
  Issuer DN: CN=CA Signing Certificate,OU=ca-1,O=domain.com Security Domain
  Not Valid Before: Tue Oct 13 20:34:36 2026
  Not Valid After: Sat Oct 13 20:34:36 2046
  Trust Flags: CTu,Cu,Cu

  Cert ID: ca_ocsp_signing
  Nickname: ocspSigningCert cert-ca-1 CA
  Token: testSC
  Serial Number: 0xe216f7bbbdb54cdf05d90db71aa32bb9
  Subject DN: CN=CA OCSP Signing Certificate,OU=ca-1,O=domain.com Security Domain
  Issuer DN: CN=CA Signing Certificate,OU=ca-1,O=domain.com Security Domain
  Not Valid Before: Tue Oct 13 20:34:49 2026
  Not Valid After: Mon Oct 02 20:34:49 2028
  Trust Flags: u,u,u

  Cert ID: sslserver
  Nickname: Server-Cert cert-ca-1
  Token: testSC
  Serial Number: 0x37043749d9d02a5a93b04afe550b28a5
  Subject DN: CN=pki.domain.com,OU=ca-1,O=domain.com Security Domain
  Issuer DN: CN=CA Signing Certificate,OU=ca-1,O=domain.com Security Domain
  Not Valid Before: Tue Oct 13 20:35:04 2026
  Not Valid After: Mon Oct 02 20:35:04 2028
  Trust Flags: u,u,u

  Cert ID: subsystem
  Nickname: subsystemCert cert-ca-1
  Token: testSC
  Serial Number: 0x38294671b44b6bd3e626699ee0b25f6a
  Subject DN: CN=Subsystem Certificate,OU=ca-1,O=domain.com Security Domain
  Issuer DN: CN=CA Signing Certificate,OU=ca-1,O=domain.com Security Domain
  Not Valid Before: Tue Oct 13 20:35:19 2026
  Not Valid After: Mon Oct 02 20:35:19 2028
  Trust Flags: u,u,u

  Cert ID: ca_audit_signing
  Nickname: auditSigningCert cert-ca-1 CA
  Token: testSC
  Serial Number: 0xe964d7f84b22933facff0a015f435a44
  Subject DN: CN=CA Audit Signing Certificate,OU=ca-1,O=domain.com Security Domain
  Issuer DN: CN=CA Signing Certificate,OU=ca-1,O=domain.com Security Domain
  Not Valid Before: Tue Oct 13 20:35:34 2026
  Not Valid After: Mon Oct 02 20:35:34 2028
  Trust Flags: u,u,Pu
```

4. List the keys in the HSM:

```
% nfkminfo -k

Key list - 10 keys
 AppName pkcs11               Ident umedb3d45a28e5a6b22b033684ce589d9e198272c2-
2ab64e740db29dbb9d04861f0598cf8c76e2a8d5
 AppName pkcs11               Ident ucedb3d45a28e5a6b22b033684ce589d9e198272c2-
754c792a76eca7fdf47817e4e74909701a68562d
 AppName pkcs11               Ident umedb3d45a28e5a6b22b033684ce589d9e198272c2-
ae962c4c96cb108dc02f42d25ee8235bbb023c97
 AppName pkcs11               Ident umedb3d45a28e5a6b22b033684ce589d9e198272c2-
17c72cdb1633e7a48cfe8fc5f69180207590d0e7
 AppName pkcs11               Ident ucedb3d45a28e5a6b22b033684ce589d9e198272c2-
178aed468a4117359991d6e997c5461d8be64202
 AppName pkcs11               Ident ucedb3d45a28e5a6b22b033684ce589d9e198272c2-
0f95a22faeee7d8cc72824da91c49c9adcea097a
 AppName pkcs11               Ident umedb3d45a28e5a6b22b033684ce589d9e198272c2-
7c0ce91f4aa02d6c303eee1676ddea2840ac492f
 AppName pkcs11               Ident ucedb3d45a28e5a6b22b033684ce589d9e198272c2-
06c390b93003a4e524a7a1675f48f272c78d352d
 AppName pkcs11               Ident umedb3d45a28e5a6b22b033684ce589d9e198272c2-
2df90386487d1ba092d1ba732d650dc386631ca8
 AppName pkcs11               Ident ucedb3d45a28e5a6b22b033684ce589d9e198272c2-
b27910037cb4446c2dbf1edc38bfa361a8d4f33b

% rocs
'rocs' key recovery tool
Useful commands: 'help', 'help intro', 'quit'.
rocs> list keys
  No. Name                     App        Protected by
    1 subsystemCert cert-ca-1  pkcs11     module (testOCS)
    2 Id: ucedb3d45a28e5a6b22b pkcs11     testOCS
    3 auditSigningCert cert-ca pkcs11     module (testOCS)
    4 caSigningCert cert-ca-1  pkcs11     module (testOCS)
    5 Id: ucedb3d45a28e5a6b22b pkcs11     testOCS
    6 Id: ucedb3d45a28e5a6b22b pkcs11     testOCS
    7 ocspSigningCert cert-ca- pkcs11     module (testOCS)
    8 Id: ucedb3d45a28e5a6b22b pkcs11     testOCS
    9 Server-Cert cert-ca-1    pkcs11     module (testOCS)
   10 Id: ucedb3d45a28e5a6b22b pkcs11     testOCS
```

5. Show the keys and their types:

```
% sudo certutil -K -d /var/lib/pki/ca-1/conf/alias -h testSC

certutil: Checking token "testSC" in slot "testSC"
Enter Password or Pin for "testSC":
< 0> mldsa    0d042533ce18dfa507dcb14c2a138b97e62a6d3c   testSC:caSigningCert cert-ca-1 CA
< 1> mldsa    0e047b7e72386d949eadb38cada850008a6ef925   testSC:ocspSigningCert cert-ca-1 CA
< 2> rsa      ed81516b98f6987dd0676b6cdb15589c09c3b373   testSC:Server-Cert cert-ca-1
< 3> mldsa    351f7282fae4138eda8e5d603017cbfe06791c24   testSC:subsystemCert cert-ca-1
< 4> mldsa    6c8e384a7fa533d6e356fe7ba79b08b617db89d9   testSC:auditSigningCert cert-ca-1 CA
```

2.4.6.3. Create a subordinate or issuing CA

Set `pki_external=True` for the `pkispawn.cfg` file so that `pkispawn` does not self-sign the `caSigningCert` object, but instead creates a PKCS #10

> certificate request file for an external root CA to sign.

1. Set the `pki_external_step_two` parameter to false in the `pkispawn.cfg` file.
2. Run `pkispawn` phase 1:

```
pkispawn -s CA --debug -f /path/to/pkispawn.cfg --skip-installation
```

3. Sign the certificate request generated at the end of phase 1, and put it in `/etc/pki/<instance>/alias/<instance>_caSigningCert.cer`.
4. Put the CA chain PKCS #7 in `/etc/pki/<instance>/alias/caChain.p7c`.

   This file should not include the new subordinate CA's certificate, only the signing CA hierarchy.

5. Change the `pki_external_step_two` parameter to true in `pkispawn.cfg`.
6. Run `pkispawn` phase 2.

   If you are using OCS-protected keys and K>1, use the preload command with `pkispawn`:

```
preload -m <module number> -f "<preload FilePath>" \
  --cardset-name=<OCS Cardset-Name> \
  pkispawn -s CA --debug -f /path/to/pkispawn.cfg
```

   Insert the OCS cards and, if prompted, enter the OCS passphrase.

   For other OCS scenarios and other protection methods, run `pkispawn` on its own:

```
pkispawn -s CA --debug -f /path/to/pkispawn.cfg
```

2.4.6.4. Configure the CA and CRL profiles

See Configure pkispawn.

Restart the CA service after modifying profiles or modifying `/etc/pki/<instance>/ca/CS.cfg`.

2.4.6.5. RHCS instance control commands

If you are using OCS-protected keys and K>1:

1. Open another terminal.
2. Run the following command:

```
preload -m <module number> -f "<preload FilePath>" \
  --cardset-name=<OCS Cardset-Name> pause
```

3. Insert the OCS cards and, if prompted, enter the OCS passphrase.
4. Leave the preload command paused until you have finished running the control commands that you intend to run, for example `systemctl start` or `systemctl restart`. This might take a minute or two. Before you cancel the preload command, check the CA web page to ensure that the system is up and running.

Function	Command
Enable on boot	Do not use. The service fails because it needs the OCS passphrase to unlock the CA private signing key.
Start manually	`systemctl start pki-tomcatd@<instance>`
Stop manually	`systemctl stop pki-tomcatd@<instance>`

Check status	`systemctl status pki-tomcatd@<instance>`
	
	`pkidaemon status <instance>`

2.5. Import the CA chain and user credentials into Firefox

Launch Firefox.

If this is a subordinate CA, manually import the external certificates in the CA chain:

1. Navigate to the drop-down menu in the upper-right corner (three horizontal bars).
2. Navigate to **Preferences > Privacy and Security**.
3. Scroll down to **Certificates**, and select **View Certificates**.
4. On the **Authorities** tab, select **Import** to import the CA chain certificates into the store.

   Repeat for each certificate in the external trust chain.

5. Configure certificate trust:
   a. If prompted during import, select all the options to trust the certificate for various activities.
   b. On the **Authorities** tab, find your CA certificate in the list organized by the organization name specified during installation. Select **Edit Trust**, select all of the options to trust the root certificate, and select **OK**.

c. Repeat for each certificate that was imported.

6. Select **OK**.

To import the CA's own chain, for both root and sub CA types:

1. Navigate to the **Retrieval** tab on the end-entity page.
2. Select the **Import CA Certificate Chain** link.
3. Select **Import the CA certificate chain into your browser**, and select **Submit**.
4. Select all of the trust options, and select **OK**.

To import the default agent's credential for both root and sub CA types:

1. Navigate to the drop-down menu in the upper-right corner (three horizontal bars).
2. Navigate to **Preferences > Privacy and Security**.
3. Scroll down to **Certificates**, and select **View Certificates**.
4. On the **Your Certificates** tab, select **Personal**, then select **Import**.
5. Navigate to the folder in which your default agent credential was created, for example `/etc/pki/<instance>/agent_alias`, and select the admin PKCS #12 file that is located there.
6. Enter the PKCS #12 passphrase defined in the `pkispawn` configuration and select **OK**.
7. Select **OK** to close **View Certificates**.

2.6. Basic system tests

2.6.1. CA application URLs

The following table lists the application URLs, using *pki.domain.com* as an example:

Web Site	URL
Unsecure URL	http://<pki.domain.com>:8080/ca/ee/ca
Secure Agent URL	https://<pki.domain.com>:8443/ca/agent/ca
Secure EE URL	https://<pki.domain.com>:8443/ca/ee/ca
Secure Admin URL	https://<pki.domain.com>:8443/ca/services

2.6.2. Test the CA functionality

1. Open a web browser from the CA.
2. Open the **Secure Agent URL**.
3. Choose the appropriate agent certificate, then select **OK**.
4. Enter the Firefox security database password if it is configured.
5. Navigate to **Update Revocation List**.
6. Ensure that **Issuing Point** is set to **MasterCRL**.
7. Select the appropriate **Signature Algorithm**. The default is likely to change when you configure `/etc/pki/<instance>/ca/CS.cfg`, but for testing, select the algorithm that you plan to use in the final configuration.
8. Create a new CRL:
    a. Select **Update**.
    b. When the **Certification Revocation List Update** has been scheduled, check the CS logs for the results.
9. Select the link on the left for **Display Revocation List**.
10. Ensure that **Issuing Point** is set to **MasterCRL**.
11. Ensure that **Display Type** is set to **Entire CRL**.
12. Select **Display** to view the new CRL.
13. To verify the CRL, check that its timestamp is within the last minute.

Chapter 3. Additional Testing

The tests in this section verify that the RHCS CA uses the nShield HSM for all of its private-key operations. They use command-line tools instead of Firefox.

Each test describes what is tested, how it relates to the HSM, the commands to run, an example of the expected output, and the criteria for the test to pass. Run the tests in order, because later tests use the files and values created by earlier tests.

Test	Description	Required
Test 1	The HSM and Security World are usable.	Yes
Test 2	The CA service is running and serving HTTPS.	Yes
Test 3	The CA system keys are stored on the HSM token.	Yes
Test 4	Agent login with the admin certificate.	Yes
Test 5	Issue a certificate signed by the HSM-protected CA key.	Yes
Test 6	OCSP reports the new certificate as good.	Yes
Test 7	Revoke the certificate.	Yes
Test 8	OCSP reports the certificate as revoked.	Yes
Test 9	The certificate record is stored in RHDS.	Yes
Test 10	Generate and verify a new CRL.	Yes
Test 11	Verify the signed audit log.	Only if audit log signing is enabled
Test 12	CA restart and unattended HSM token login.	Optional
Test 13	The CA cannot sign while the HSM is unavailable (expected failure).	Optional

Test 12 and Test 13 restart the CA without user interaction. Do not run them if the CA keys are protected by an OCS with K>1, because the CA can then only start under `preload` with the cards inserted.

3.1. Before you begin

Make sure that:

- The CA instance was deployed as described in Procedures. The examples in this section use the instance name `ca-1`, the Softcard `testSC`, and the host name `pki.domain.com`.
- You can log in to the CA server as a user with `sudo` privileges.
- You know the admin PKCS #12 password (`pki_client_pkcs12_password` in the `pkispawn` configuration file).
- If the CA keys are protected by an OCS, the OCS is inserted in the HSM.

3.1.1. Prepare the test environment

Run all the commands, in this section and in the tests, in the same shell session on the CA server.

1. Create and navigate to a private working folder:

   ```
   mkdir -m 700 ~/hsm-test
   cd ~/hsm-test
   export PATH=$PATH:/opt/nfast/bin
   ```

   All the files created during the tests are saved in this folder. Some of them contain secrets. Delete the folder at the end of the tests, as described in Clean up.

2. Set the following shell variables. Replace `ca-1` and `testSC` with your CA instance name and HSM token name:

   ```
   INSTANCE=ca-1
   TOKEN=testSC
   CS_CFG=/var/lib/pki/$INSTANCE/ca/conf/CS.cfg
   PW_CONF=/var/lib/pki/$INSTANCE/conf/password.conf
   ALIAS_DIR=/var/lib/pki/$INSTANCE/conf/alias
   HOST=$(sudo grep -m1 '^machineName=' $CS_CFG | cut -d= -f2-)
   BASEDN=$(sudo grep -m1 '^internaldb.basedn=' $CS_CFG | cut -d= -f2-)
   HTTPS_URL=https://$HOST:8443
   HTTP_URL=http://$HOST:8080
   echo "$HOST $BASEDN"
   ```

   Example output:

   ```
   pki.domain.com o=pki-ca-1-CA
   ```

3. Define the following shell functions:

   ```
   # Total number of commands processed by the hardserver.
   hsm_cmds() { stattree | awk '/CmdCount/ && $NF ~ /^[0-9]+$/ {s += $NF} END {print s}'; }

   # Waits until the CA reports that it is running.
   wait_for_ca() {
   ```

```
  until curl -sk --max-time 10 $HTTPS_URL/ca/admin/ca/getStatus \
      | grep -q '"Status" *: *"running"'; do
    sleep 5
  done
  echo "CA is running"
}
```

`hsm_cmds` reads the hardserver command counters. Tests 5, 6, 8, and 10 compare the counter before and after each signing operation, as evidence that the operation was carried out by the HSM. The counter is evidence only, not a pass or fail criterion.

4. Check that the hardserver and the CA are running:

```
enquiry | grep -m2 mode
curl -sk $HTTPS_URL/ca/admin/ca/getStatus
```

Both lines from `enquiry` must show `operational`, and the CA must report `"Status" : "running"`.

If `enquiry` fails, start the hardserver, then restart the CA so it opens new sessions with the HSM:

```
sudo /opt/nfast/sbin/init.d-ncipher start
sudo systemctl restart pki-tomcatd@$INSTANCE
wait_for_ca
```

5. Record the token protection. For a Softcard, the token name is listed by `nfkminfo -s`. For an OCS, it is listed by `nfkminfo -c`, together with its K/N quorum:

```
nfkminfo -s
nfkminfo -c
```

If the token is an OCS with K>1, check that `/opt/nfast/cknfastrc` contains `NFAST_NFKM_TOKENSFILE`, and do not run the following tests:

   - Test 12: CA restart and unattended HSM token login
   - Test 13: The CA cannot sign while the HSM is unavailable (expected failure).

6. Check which entries the CA password file contains, without displaying their values:

```
sudo ls -l $PW_CONF
sudo cut -d= -f1 $PW_CONF
```

Example output:

```
-rw-rw----. 1 pkiuser pkiuser 96 Oct 23 18:40 /var/lib/pki/ca-1/conf/password.conf
hardware-testSC
```

```
internal
internaldb
replicationdb
```

The CA reads this file at every start-up to log in to its tokens:

- `hardware-<token>` is the PIN of the HSM token.
- `internal` is the password of the CA's NSS database.
- `internaldb` is the password the CA uses to bind to RHDS.

3.1.2. (Optional) Enable signed audit logging

Test 11: Verify the signed audit log verifies the CA audit log, which is signed with the audit signing key on the HSM. It can only run if audit log signing is enabled.

To ensure that audit log signing is enabled before you start the tests:

1. Check the current audit signing settings:

```
sudo grep -E \
  '^log.instance.SignedAudit.(logSigning|signedAuditCertNickname|flushInterval)=' \
  $CS_CFG
```

If the output contains `log.instance.SignedAudit.logSigning=true`, audit log signing is already enabled. Skip the rest of this procedure.

2. Back up `CS.cfg`:

```
AUDIT_BACKUP=$CS_CFG.bak.$(date +%Y%m%d%H%M%S)
sudo cp -p $CS_CFG $AUDIT_BACKUP
```

3. Stop the CA, so it does not overwrite your changes to `CS.cfg`:

```
sudo systemctl stop pki-tomcatd@$INSTANCE
```

4. If `signedAuditCertNickname` is not set, set it to the audit signing certificate on the HSM token:

```
sudo pki-server ca-config-set -i $INSTANCE \
  log.instance.SignedAudit.signedAuditCertNickname \
  "$TOKEN:auditSigningCert cert-$INSTANCE CA"
```

5. Enable audit log signing:

```
sudo pki-server ca-config-set -i $INSTANCE \
```

```
  log.instance.SignedAudit.logSigning true
```

6. Record the current length of the audit log. The existing entries were written unsigned, so Test 11: Verify the signed audit log only verifies the entries written after this point:

```
AUDIT_START=$(sudo wc -l /var/log/pki/$INSTANCE/ca/signedAudit/ca_audit \
  | awk '{print $1}')
echo $AUDIT_START
```

7. Start the CA and wait until it is running:

```
sudo systemctl start pki-tomcatd@$INSTANCE
wait_for_ca
```

8. Confirm the new settings:

```
sudo grep -E \
  '^log.instance.SignedAudit.(logSigning|signedAuditCertNickname)=' \
  $CS_CFG
```

At the end of the tests, restore the original configuration as described in Clean up.

3.2. Test 1: The HSM and Security World are usable

What is tested

The HSM module is operational, and the Security World and module #1 are in a usable state.

HSM relevance

The CA can only use its keys if the hardserver can reach module #1 and the Security World is loaded.

Prerequisites

- The nShield software is installed and the hardserver is running.
- The HSM is enrolled with this server.
- The Security World is loaded on the HSM.

Steps

1. Check the HSM module mode:

```
enquiry
```

Example output (excerpt):

```
...
Module #1:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        1234-5678-9ABC
 mode                 operational
 version              13.8.5
...
 module type          nShield Connect XC
...
 connection status    OK
...
 hardware status      OK
```

2. Check the Security World and module state:

```
nfkminfo
```

Example output (excerpt):

```
World
 generation  2
 state       0x37270009 Initialised Usable Recovery !PINRecovery !ExistingClient RTC NVRAM FTO
  AlwaysUseStrongPrimes !DisablePKCS1Padding !PpStrengthCheck AuditLogging !StrictKeyAgreement
  SEEDebug
...
Module #1
 generation 2
 state      0x2 Usable
...
```

3. Measure the idle HSM command rate. This is the baseline for the HSM command counters in later tests:

```
C1=$(hsm_cmds)
sleep 5
C2=$(hsm_cmds)
IDLE=$((C2 - C1))
echo "Idle baseline: $IDLE commands in 5 seconds"
```

Example output:

```
Idle baseline: 1258 commands in 5 seconds
```

Result

The test passes if `enquiry` shows `mode operational` for module #1, and `nfkminfo` shows `Usable` in the state of both the World and module #1.

3.3. Test 2: The CA service is running and serving HTTPS

What is tested

The `pki-tomcatd` service is active, and the CA answers its status request over HTTPS.

HSM relevance

At start-up, the CA logs in to the HSM token with the PIN in `password.conf`. The HTTPS handshake is signed with the SSL server key, which is stored on the HSM, so a successful TLS connection shows that the HSM key is in use.

Prerequisites

- Test 1: The HSM and Security World are usable passed.
- The CA instance was installed with `pkispawn` on the HSM token.
- The token PIN is stored in `password.conf` (`hardware-<token>=`).

Steps

1. Check that the CA service is running:

```
sudo systemctl is-active pki-tomcatd@$INSTANCE
```

   Expected output:

```
active
```

2. Record the CA state, ports, and URLs:

```
sudo pki-server status $INSTANCE
```

   Example output (excerpt):

```
  Instance ID: ca-1
  Active: True
  Unsecure Port: 8080
  Secure Port: 8443
...
  CA Subsystem:
...
    Unsecure URL:        http://pki.domain.com:8080/ca/ee/ca
    Secure Agent URL:    https://pki.domain.com:8443/ca/agent/ca
    Secure EE URL:       https://pki.domain.com:8443/ca/ee/ca
    Secure Admin URL:    https://pki.domain.com:8443/ca/services
```

3. Export the CA signing certificate. Later tests use it to verify everything that the CA signs:

```
sudo pki-server cert-export ca_signing -i $INSTANCE \
  --cert-file ca_signing_cert.pem
```

4. Connect to the CA over TLS, verifying the server certificate against the CA certificate:

```
curl -sS --max-time 30 --cacert ca_signing_cert.pem \
  $HTTPS_URL/ca/admin/ca/getStatus
```

Example output:

```
{
  "Response" : {
    "State" : "1",
    "Type" : "CA",
    "Status" : "running",
    "Version" : "11.10.1",
    "ProductVersion" : "Red Hat Certificate System 11.0.0"
  }
}
```

Result

The test passes if the service is `active` and `getStatus` returns `"Status" : "running"` over a verified TLS connection. If the test fails, check `journalctl -u pki-tomcatd@<instance>` for token login errors.

3.4. Test 3: The CA system keys are stored on the HSM token

What is tested

All CA system certificates have their private keys on the HSM token, not in the internal (software) NSS token.

HSM relevance

This shows that the keys were generated in, and are protected by, the HSM rather than stored in software.

Prerequisites

- Test 2: The CA service is running and serving HTTPS passed.
- The HSM token PIN (`hardware-<token>=`) and the internal token password (`internal=`) are in `password.conf`.

Steps

1. Copy the token PINs from `password.conf` into files in the working folder, without displaying them:

```
sudo grep -m1 "^hardware-$TOKEN=" $PW_CONF \
  | cut -d= -f2- | tr -d '\n' > hsm_token_pin.txt
sudo grep -m1 '^internal=' $PW_CONF \
  | cut -d= -f2- | tr -d '\n' > internal_token_pin.txt
```

2. List the CA system certificates:

```
sudo pki-server cert-find -i $INSTANCE
```

Example output (excerpt):

```
  Cert ID: ca_signing
  Nickname: caSigningCert cert-ca-1 CA
  Token: testSC
...
  Cert ID: ca_ocsp_signing
  Nickname: ocspSigningCert cert-ca-1 CA
  Token: testSC
...
  Cert ID: sslserver
  Nickname: Server-Cert cert-ca-1
  Token: testSC
...
  Cert ID: subsystem
  Nickname: subsystemCert cert-ca-1
  Token: testSC
...
  Cert ID: ca_audit_signing
  Nickname: auditSigningCert cert-ca-1 CA
  Token: testSC
...
```

Note the nickname of each certificate.

3. List the private keys on the HSM token:

```
sudo certutil -K -d $ALIAS_DIR -h $TOKEN -f hsm_token_pin.txt
```

Example output:

```
certutil: Checking token "testSC" in slot "testSC"
< 0> mldsa    64b9452a89acf1fe5be2c6ce8d4f3ebe087dad42   testSC:caSigningCert cert-ca-1 CA
< 1> mldsa    ae1b3594762344bd41a3471b64a370a391beb3ad   testSC:ocspSigningCert cert-ca-1 CA
< 2> rsa      959d612696c69b633d159aea8b147435a5b31d1e   testSC:Server-Cert cert-ca-1
< 3> mldsa    091c91c5a34160a6580f48753846e3bcef074e19   testSC:subsystemCert cert-ca-1
< 4> mldsa    3b78e5e2b3d285563b5b5ac3b9f2a45bd5400d30   testSC:auditSigningCert cert-ca-1 CA
```

4. List the private keys in the internal software token:

```
sudo certutil -K -d $ALIAS_DIR -f internal_token_pin.txt
```

Expected output:

```
certutil: Checking token "NSS Certificate DB" in slot "NSS User Private Key and Certificate Services"
certutil: no keys found
```

`certutil` exits with code 255 when a token holds no private keys. This is the expected result for the internal token.

Result

The test passes if every system certificate nickname from `pki-server cert-find` is listed on the HSM token, and none are listed in the internal software token.

3.5. Test 4: Agent login with the admin certificate

What is tested

A client NSS database is created, the CA certificate is trusted, the admin PKCS #12 is imported, and the agent interface is accessed with the admin certificate. This replaces importing the CA chain and the agent credential into Firefox.

HSM relevance

The CA certificate trusted here is signed by the CA signing key on the HSM, and the TLS session uses the SSL server key on the HSM.

Prerequisites

- Test 2: The CA service is running and serving HTTPS passed, and `ca_signing_cert.pem` is in the working folder.
- The admin PKCS #12 file created by `pkispawn` is at `/root/.dogtag/<instance>/ca_admin_cert.p12`.

Steps

1. Save the admin PKCS #12 password in a file. Enter the password when prompted:

```
read -rs -p "Admin PKCS #12 password: " P12PW; echo
printf '%s' "$P12PW" > admin_p12_password.txt
unset P12PW
```

2. Create a random password for the client NSS database:

```
head -c 24 /dev/urandom | base64 | tr -d '\n' > client_nssdb_password.txt
```

3. Copy the admin PKCS #12 file, which only `root` can read, to the working folder:

```
sudo cat /root/.dogtag/$INSTANCE/ca_admin_cert.p12 > admin_cert_and_key.p12
```

4. Define the common `pki` client options (server URL, client NSS database, and its password file):

```
PKI="pki -U $HTTPS_URL -d client_nssdb -C client_nssdb_password.txt"
```

5. Create the client NSS database, which takes the place of Firefox:

```
$PKI client-init --force
```

6. Import and trust the CA signing certificate:

```
$PKI client-cert-import "CA Certificate" --ca-cert ca_signing_cert.pem
```

   Expected output:

```
Imported certificate "CA Certificate"
```

7. Import the admin certificate and key:

```
$PKI pkcs12-import --pkcs12 admin_cert_and_key.p12 \
  --password-file admin_p12_password.txt
```

8. Find the nickname of the imported admin certificate. It is the certificate with the trust attributes `u,u,u`:

```
certutil -L -d client_nssdb
```

   Example output:

```
Certificate Nickname                                         Trust Attributes
                                                             SSL,S/MIME,JAR/XPI

CA Certificate                                               CT,C,C
PKI Administrator for domain.com                             u,u,u
```

9. Save the nickname in a variable:

```
ADMIN_NICK="PKI Administrator for domain.com"
```

10. Call an agent-only function with the admin certificate. Replace `caadmin` with the value of `pki_admin_uid` in the `pkispawn` configuration file if you changed it:

```
$PKI -n "$ADMIN_NICK" ca-user-show caadmin
```

Example output:

```
--------------
User "caadmin"
--------------
  User ID: caadmin
  Full name: caadmin
  Email: caadmin@domain.com
  Type: adminType
  State: 1
```

11. Save the admin certificate and private key as PEM files. Test 10: Generate and verify a new CRL uses them with `curl`, which cannot read the client NSS database:

```
openssl pkcs12 -in admin_cert_and_key.p12 \
  -passin file:admin_p12_password.txt \
  -clcerts -nokeys -out admin_cert.pem
openssl pkcs12 -in admin_cert_and_key.p12 \
  -passin file:admin_p12_password.txt \
  -nocerts -nodes -out admin_private_key.pem
chmod 600 admin_private_key.pem
```

`admin_private_key.pem` contains the admin private key unencrypted. Delete it at the end of the tests, as described in Clean up.

Result

The test passes if `ca-user-show` returns the admin user over the client-authenticated HTTPS connection. If it fails, check the PKCS #12 password and that the PKCS #12 file exists.

3.6. Test 5: Issue a certificate signed by the HSM-protected CA key

What is tested

A new certificate request is submitted, approved by the agent, and issued. The issued certificate is verified against the CA certificate.

HSM relevance

Issuing a certificate requires a signature with the CA signing key, which only exists on the HSM.

Prerequisites

- Test 4: Agent login with the admin certificate passed.
- The `caUserCert` enrollment profile is enabled on the CA.

Steps

1. Choose a unique name for the test user and record the HSM command counter:

```
TEST_UID=hsmtest-$(date +%Y%m%d%H%M%S)
BEFORE=$(hsm_cmds)
```

2. Create a key pair and a certificate request for the test user:

```
$PKI nss-cert-request --subject "UID=$TEST_UID" \
  --ext /usr/share/pki/tools/examples/certs/testuser.conf \
  --csr test_user_request.csr
```

The test user's key pair is generated in the client NSS database, in software. Only the CA's signature on the certificate uses the HSM. If `/usr/share/pki/tools/examples/certs/testuser.conf` does not exist, omit the `--ext` option.

3. Submit the request to the CA:

```
$PKI ca-cert-request-submit --profile caUserCert \
  --csr-file test_user_request.csr \
  --subject "UID=$TEST_UID"
```

Example output:

```
----------------------------
Submitted certificate request
----------------------------
  Request ID: 0xe1828106f8d1ff5e765153d9669b74c0
  Type: enrollment
  Request Status: pending
  Operation Result: success
...
```

4. Approve the request as agent. Replace `<request ID>` with the `Request ID` from the previous step:

```
$PKI -n "$ADMIN_NICK" ca-cert-request-approve <request ID> --force
```

Example output:

```
---------------------------------------------------------------
Approved certificate request 0xe1828106f8d1ff5e765153d9669b74c0
---------------------------------------------------------------
  Request ID: 0xe1828106f8d1ff5e765153d9669b74c0
  Type: enrollment
  Request Status: complete
  Operation Result: success
  Certificate ID: 0xe67dd0bdd4662bf3989553cadf1ca379
...
```

The CA signs the certificate with its HSM key when the request is approved.

5. Record the HSM command counter and save the certificate serial number. Replace `<certificate ID>` with the `Certificate ID` from the previous step:

```
AFTER=$(hsm_cmds); echo "HSM commands: $((AFTER - BEFORE)) (idle baseline: $IDLE in 5 seconds)"
SERIAL=<certificate ID>
```

Example output:

```
HSM commands: 2077 (idle baseline: 1258 in 5 seconds)
```

6. Download the issued certificate:

```
$PKI ca-cert-export $SERIAL --output-file test_user_cert.pem
```

7. Display the subject, issuer, serial number, and key usage of the certificate:

```
openssl x509 -in test_user_cert.pem -noout \
  -subject -issuer -serial -ext keyUsage
```

Example output:

```
subject=UID=hsmtest-20261023202949
issuer=O=domain.com Security Domain, OU=ca-1, CN=CA Signing Certificate
serial=E67DD0BDD4662BF3989553CADF1CA379
X509v3 Key Usage: critical
    Digital Signature, Non Repudiation, Key Encipherment
```

8. Display the algorithm that the CA's HSM key signed with:

```
openssl x509 -in test_user_cert.pem -noout -text | grep -m1 'Signature Algorithm'
```

Example output:

```
        Signature Algorithm: ML-DSA-44
```

9. Verify the certificate signature against the CA certificate:

```
openssl verify -CAfile ca_signing_cert.pem test_user_cert.pem
```

Expected output:

```
test_user_cert.pem: OK
```

Result

The test passes if the request is approved, a certificate is issued, and `openssl verify` reports `OK`. An increase in the HSM command counter well above the idle baseline is further evidence that the HSM signed the certificate.

3.7. Test 6: OCSP reports the new certificate as good

What is tested

The CA's OCSP responder is queried for the new certificate before it is revoked.

HSM relevance

The OCSP response is signed with the OCSP signing key on the HSM, and `openssl` verifies that signature.

Prerequisites

- Test 5: Issue a certificate signed by the HSM-protected CA key passed.
- The OCSP responder is reachable on port 8080.

Steps

1. Query the OCSP responder, and record the HSM command counter before and after:

```
BEFORE=$(hsm_cmds)
openssl ocsp -issuer ca_signing_cert.pem -cert test_user_cert.pem \
  -CAfile ca_signing_cert.pem -url $HTTP_URL/ca/ocsp
AFTER=$(hsm_cmds); echo "HSM commands: $((AFTER - BEFORE)) (idle baseline: $IDLE in 5 seconds)"
```

Example output:

```
Response verify OK
test_user_cert.pem: good
    This Update: Oct 23 20:29:59 2026 GMT
```

```
HSM commands: 1267 (idle baseline: 1258 in 5 seconds)
```

Result

The test passes if the output contains `Response verify OK` and the certificate status is `good`. A verification failure points to the OCSP signing key or its certificate.

3.8. Test 7: Revoke the certificate

What is tested

The agent revokes the test certificate with the reason `Key_Compromise`.

HSM relevance

The agent request uses a client-authenticated TLS session with the SSL server key on the HSM. The revocation is recorded in the CA database.

Prerequisites

- Test 4: Agent login with the admin certificate and Test 5: Issue a certificate signed by the HSM-protected CA key passed.

Steps

1. Revoke the test certificate as agent:

```
$PKI -n "$ADMIN_NICK" ca-cert-revoke $SERIAL \
  --reason Key_Compromise --force
```

Example output:

```
---------------------------------------------------------
Revoked certificate "0xe67dd0bdd4662bf3989553cadf1ca379"
---------------------------------------------------------
  Serial Number: 0xe67dd0bdd4662bf3989553cadf1ca379
  Subject DN: UID=hsmtest-20261023202949
  Issuer DN: CN=CA Signing Certificate,OU=ca-1,O=domain.com Security Domain
  Status: REVOKED
...
  Revoked On: Fri Oct 23 20:30:02 UTC 2026
  Revoked By: caadmin
```

2. Read the certificate status back from the CA:

```
$PKI ca-cert-show $SERIAL
```

Example output:

```
  Serial Number: 0xe67dd0bdd4662bf3989553cadf1ca379
```

```
  Subject DN: UID=hsmtest-20261023202949
  Issuer DN: CN=CA Signing Certificate,OU=ca-1,O=domain.com Security Domain
  Status: REVOKED
...
```

Result

The test passes if `ca-cert-show` reports `Status: REVOKED`.

3.9. Test 8: OCSP reports the certificate as revoked

What is tested

The OCSP responder is queried again after the revocation.

HSM relevance

A new OCSP response is signed with the OCSP signing key on the HSM.

Prerequisites

- Test 7: Revoke the certificate passed.
- The OCSP responder is reachable on port 8080.

Steps

1. Query the OCSP responder, and record the HSM command counter before and after:

```
BEFORE=$(hsm_cmds)
openssl ocsp -issuer ca_signing_cert.pem -cert test_user_cert.pem \
  -CAfile ca_signing_cert.pem -url $HTTP_URL/ca/ocsp
AFTER=$(hsm_cmds); echo "HSM commands: $((AFTER - BEFORE)) (idle baseline: $IDLE in 5 seconds)"
```

Example output:

```
Response verify OK
test_user_cert.pem: revoked
    This Update: Oct 23 20:30:05 2026 GMT
    Revocation Time: Oct 23 20:30:02 2026 GMT
HSM commands: 1268 (idle baseline: 1258 in 5 seconds)
```

Result

The test passes if the output contains `Response verify OK` and the certificate status is `revoked`.

3.10. Test 9: The certificate record is stored in RHDS

What is tested

The test certificate's entry exists in the CA's certificate repository in RHDS and shows the revoked status.

HSM relevance

RHDS stores the CA data, not keys. This test confirms that the HSM-signed certificate was persisted.

Prerequisites

- Test 7: Revoke the certificate passed.
- RHDS runs on the CA server, with its LDAPI socket in `/run`.
- Root autobind is enabled in RHDS.

Steps

1. Convert the certificate serial number to decimal. RHDS stores certificates under their decimal serial number:

```
DEC_SERIAL=$(python3 -c "import sys; print(int(sys.argv[1], 16))" $SERIAL)
echo $DEC_SERIAL
```

Example output:

```
306375709918902119035888128361431343993
```

2. Find the RHDS LDAPI socket:

```
ls /run/slapd-*.socket
```

Example output:

```
/run/slapd-ca-1.socket
```

3. Look up the certificate record in RHDS. In the `ldapi://` URL, each `/` of the socket path is written as `%2F`:

```
sudo ldapsearch -Y EXTERNAL -Q -LLL \
  -H 'ldapi://%2Frun%2Fslapd-ca-1.socket' \
  -b "ou=certificateRepository,ou=ca,$BASEDN" \
  "(cn=$DEC_SERIAL)" cn certStatus
```

Example output:

```
dn: cn=306375709918902119035888128361431343993,ou=certificateRepository,ou=ca,
```

```
 o=pki-ca-1-CA
cn: 306375709918902119035888128361431343993
certStatus: REVOKED
```

Result

The test passes if `ldapsearch` finds the entry with `certStatus: REVOKED`. If `ldapsearch` fails to authenticate, root autobind might be disabled in RHDS. In that case, the CA side of the revocation was already verified in Test 7: Revoke the certificate.

3.11. Test 10: Generate and verify a new CRL

What is tested

The agent asks the CA to regenerate the MasterCRL. The CRL is downloaded, its signature is verified, its date is checked, and the revoked serial number is looked up. This replaces the **Update Revocation List** and **Display Revocation List** steps in the agent web interface.

HSM relevance

The CRL is signed with the CA signing key on the HSM.

Prerequisites

- Test 4: Agent login with the admin certificate passed, and `admin_cert.pem` and `admin_private_key.pem` are in the working folder.
- Test 7: Revoke the certificate passed.

Steps

1. Download the current CRL and note its number:

```
curl -sS --max-time 60 -o master_crl.der \
  "$HTTP_URL/ca/ee/ca/getCRL?op=getCRL&crlIssuingPoint=MasterCRL"
openssl crl -inform DER -in master_crl.der -noout -crlnumber
```

Example output:

```
crlNumber=0x0F
```

2. Ask the CA to sign a new CRL now, and record the HSM command counter before and after:

```
BEFORE=$(hsm_cmds)
curl -sS --max-time 120 --cacert ca_signing_cert.pem \
  --cert admin_cert.pem --key admin_private_key.pem \
  -d 'crlIssuingPoint=MasterCRL&waitForUpdate=true&xml=true' \
  $HTTPS_URL/ca/agent/ca/updateCRL
```

```
AFTER=$(hsm_cmds); echo; echo "HSM commands: $((AFTER - BEFORE)) (idle baseline: $IDLE in 5 seconds)"
```

Example output:

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?><xml><header><crlIssuingPoint>MasterCRL</crlIssuingPoint><time>16</time></header><fix
ed/><records/></xml>
HSM commands: 1280 (idle baseline: 1258 in 5 seconds)
```

3. Download the new CRL:

```
curl -sS --max-time 60 -o master_crl.der \
  "$HTTP_URL/ca/ee/ca/getCRL?op=getCRL&crlIssuingPoint=MasterCRL"
```

4. Verify the CRL signature against the CA certificate:

```
openssl crl -inform DER -in master_crl.der -CAfile ca_signing_cert.pem -noout
```

Expected output:

```
verify OK
```

5. Display the CRL number and issue time:

```
openssl crl -inform DER -in master_crl.der -noout \
  -crlnumber -lastupdate -nextupdate
date -u
```

Example output:

```
crlNumber=0x10
lastUpdate=Oct 23 20:30:07 2026 GMT
nextUpdate=Oct 23 21:00:00 2026 GMT
Fri Oct 23 20:30:09 UTC 2026
```

6. Display the signature algorithm and check that the revoked serial number is listed in the CRL:

```
openssl crl -inform DER -in master_crl.der -noout -text \
  | grep -m1 'Signature Algorithm'
openssl crl -inform DER -in master_crl.der -noout -text \
  | grep -i "Serial Number: ${SERIAL#0x}"
```

Example output:

```
        Signature Algorithm: ML-DSA-44
```

```
    Serial Number: E67DD0BDD4662BF3989553CADF1CA379
```

Result

The test passes if all of the following are true:

- The CRL number is higher than before the update, so a new CRL was signed.
- `openssl crl` reports `verify OK`.
- `lastUpdate` is within the last 5 minutes.
- The serial number of the certificate revoked in Test 7: Revoke the certificate is listed in the CRL.

3.12. Test 11: Verify the signed audit log

What is tested

The CA's signed audit log is verified with `AuditVerify`, using the audit signing certificate.

HSM relevance

Audit log signatures are made with the audit signing key on the HSM. Valid signatures prove that this key is in use.

Prerequisites

- Test 4: Agent login with the admin certificate passed.
- Audit log signing is enabled. See (Optional) Enable signed audit logging.
- The audit signing key is on the HSM token.

Steps

1. Check that audit log signing is enabled, and display the flush interval:

```
sudo grep -E '^log.instance.SignedAudit.(logSigning|flushInterval)=' $CS_CFG
```

   If `logSigning` is not `true`, skip this test.

2. Wait for the flush interval (in seconds) plus 5 seconds. The CA signs the audit log each time it flushes its buffer, so this ensures that the entries from the previous tests are signed.
3. Export the audit signing certificate:

```
sudo pki-server cert-export ca_audit_signing -i $INSTANCE \
  --cert-file audit_signing_cert.pem
```

4. Add the certificate to the client NSS database with the `P` trust flag, which `AuditVerify` requires. Use the audit signing certificate nickname from Test 3: The CA system keys are stored on the HSM token:

```
certutil -A -d client_nssdb -n "auditSigningCert cert-$INSTANCE CA" \
  -t ',,P' -i audit_signing_cert.pem \
  -f client_nssdb_password.txt
```

5. Copy the audit log to the working folder, because only `root` can read it:

   - If you enabled audit log signing in (Optional) Enable signed audit logging, copy only the entries written after signing was enabled:

```
mkdir -p audit_logs
sudo tail -n +$((AUDIT_START + 1)) \
  /var/log/pki/$INSTANCE/ca/signedAudit/ca_audit > audit_logs/ca_audit
printf '%s' audit_logs/ca_audit > audit_log_list.txt
```

   - If audit log signing was already enabled, copy all the audit log files:

```
mkdir -p audit_logs
for f in $(sudo ls -1tr /var/log/pki/$INSTANCE/ca/signedAudit | grep '^ca_audit'); do
  sudo cat /var/log/pki/$INSTANCE/ca/signedAudit/$f > audit_logs/$f
done
ls -1tr audit_logs/* | paste -sd, - | tr -d '\n' > audit_log_list.txt
```

   `AuditVerify` reads the list of log files to check, separated by commas, from `audit_log_list.txt`.

6. Verify the audit log signatures:

```
AuditVerify -d client_nssdb -n "auditSigningCert cert-$INSTANCE CA" \
  -a audit_log_list.txt -P '' -v
```

   The output reports the number of valid and invalid signatures.

Result

The test passes if `AuditVerify` reports at least one valid signature and no invalid signatures.

- If no signatures are found, the CA has not yet flushed the audit buffer. Wait a few minutes and run `AuditVerify` again.
- If the signatures are invalid, the log was altered or was not signed by the audit signing key.
- If `AuditVerify` reports `Unknown signing certificate key type`, this version of `AuditVerify` cannot verify signatures made with the key type of the audit signing

key, for example ML-DSA. This is a limitation of the verification tool, not of the HSM. To confirm that the CA wrote signed entries, count them:

```
grep -c AUDIT_LOG_SIGNING audit_logs/*
```

3.13. Test 12: CA restart and unattended HSM token login

What is tested

The CA service is restarted, and it must come back up and sign a new CRL.

HSM relevance

On start-up, the CA must open a session with the HSM and log in to the HSM token using the PIN in `password.conf`, without user interaction.

Prerequisites

- Test 10: Generate and verify a new CRL passed.
- The token is a Softcard, or an OCS with K=1. Do not run this test for an OCS with K>1 (`NFAST_NFKM_TOKENSFILE` set in `/opt/nfast/cknfastrc`).
- The token PIN is in `password.conf` (`hardware-<token>=`).

Steps

1. Download the current CRL and note its number:

```
curl -sS --max-time 60 -o master_crl.der \
  "$HTTP_URL/ca/ee/ca/getCRL?op=getCRL&crlIssuingPoint=MasterCRL"
openssl crl -inform DER -in master_crl.der -noout -crlnumber
```

2. Restart the CA and wait until it is running:

```
sudo systemctl restart pki-tomcatd@$INSTANCE
wait_for_ca
```

   If the CA does not report that it is running within 5 minutes, press Ctrl+C and check `journalctl -u pki-tomcatd@<instance>`. A wrong token PIN in `password.conf` or an unavailable HSM are the usual causes.

3. Ask the CA to sign a new CRL:

```
curl -sS --max-time 120 --cacert ca_signing_cert.pem \
  --cert admin_cert.pem --key admin_private_key.pem \
  -d 'crlIssuingPoint=MasterCRL&waitForUpdate=true&xml=true' \
  $HTTPS_URL/ca/agent/ca/updateCRL
```

4. Download the CRL and note its number:

```
curl -sS --max-time 60 -o master_crl.der \
  "$HTTP_URL/ca/ee/ca/getCRL?op=getCRL&crlIssuingPoint=MasterCRL"
openssl crl -inform DER -in master_crl.der -noout -crlnumber
```

Result

The test passes if the CA restarts, and the CRL number after the restart is higher than before it. If the CA restarted but no new CRL was signed, check the CA debug log for token errors.

3.14. Test 13: The CA cannot sign while the HSM is unavailable (expected failure)

What is tested

The nShield hardserver is stopped, and the CA is asked to sign a new CRL. The hardserver is then started, and the CA restarted.

HSM relevance

With the hardserver stopped, the CA has no access to the HSM, so every private-key operation (CRL signing and the TLS handshake) must fail. This proves that the CA keys exist only in the HSM.

Prerequisites

- Test 10: Generate and verify a new CRL passed.
- The token is a Softcard, or an OCS with K=1.
- No other application on this server needs the HSM during the test.

This test briefly stops the hardserver, which disconnects every application on this server from the HSM. Make sure that you complete the restore steps, even if the test is interrupted.

Steps

1. Download the current CRL and note its number:

```
curl -sS --max-time 60 -o master_crl.der \
  "$HTTP_URL/ca/ee/ca/getCRL?op=getCRL&crlIssuingPoint=MasterCRL"
openssl crl -inform DER -in master_crl.der -noout -crlnumber
```

2. Stop the hardserver, to cut the CA off from the HSM:

```
sudo /opt/nfast/sbin/init.d-ncipher stop
```

3. Ask the CA to sign a new CRL. This is expected to fail:

```
curl -sS --max-time 120 --cacert ca_signing_cert.pem \
  --cert admin_cert.pem --key admin_private_key.pem \
  -d 'crlIssuingPoint=MasterCRL&waitForUpdate=true&xml=true' \
  $HTTPS_URL/ca/agent/ca/updateCRL
```

4. Download the CRL and note its number:

```
curl -sS --max-time 60 -o master_crl.der \
  "$HTTP_URL/ca/ee/ca/getCRL?op=getCRL&crlIssuingPoint=MasterCRL"
openssl crl -inform DER -in master_crl.der -noout -crlnumber
```

   The CRL number must be the same as before the hardserver was stopped.

5. Restore the HSM connection:

```
sudo /opt/nfast/sbin/init.d-ncipher start
enquiry | grep -m2 mode
```

   Both lines from `enquiry` must show `operational`.

6. Restart the CA so it opens new sessions with the HSM, and wait until it is running:

```
sudo systemctl restart pki-tomcatd@$INSTANCE
wait_for_ca
```

7. Ask the CA to sign a new CRL, and check that the CRL number increases:

```
curl -sS --max-time 120 --cacert ca_signing_cert.pem \
  --cert admin_cert.pem --key admin_private_key.pem \
  -d 'crlIssuingPoint=MasterCRL&waitForUpdate=true&xml=true' \
  $HTTPS_URL/ca/agent/ca/updateCRL
curl -sS --max-time 60 -o master_crl.der \
  "$HTTP_URL/ca/ee/ca/getCRL?op=getCRL&crlIssuingPoint=MasterCRL"
openssl crl -inform DER -in master_crl.der -noout -crlnumber
```

Result

The test passes if both of the following are true:

- No new CRL was signed while the hardserver was stopped.
- After the hardserver was started and the CA restarted, the CA signed a new CRL.

  If a CRL was signed while the hardserver was stopped, not all CA keys are on the HSM. Check Test 3: The CA system keys are stored on the HSM token. If the CA

did not recover after the hardserver was started, check `journalctl -u pki-tomcatd@<instance>` and the output of `enquiry`.

3.15. Clean up

1. If you enabled audit log signing in (Optional) Enable signed audit logging, restore the original `CS.cfg`:

```
sudo systemctl stop pki-tomcatd@$INSTANCE
sudo cp -p $AUDIT_BACKUP $CS_CFG
sudo systemctl start pki-tomcatd@$INSTANCE
wait_for_ca
sudo grep -E '^log.instance.SignedAudit.logSigning=' $CS_CFG
```

   The CA is stopped first, so it does not rewrite `CS.cfg` while it is restored.

2. Delete the working folder. It contains the token PINs, the admin PKCS #12 password, and the unencrypted admin private key:

```
cd ~
rm -rf ~/hsm-test
```

Chapter 4. Configure pkispawn

4.1. Modifying the sample pkispawn configuration file

4.1.1. Hostnames

Set these values as appropriate for your system. If you run multiple instances on the same server, it is not recommended to use the system FQDN as the RHCS server hostname.

- RHCS server: `pki_hostname=pki.domain.com`
- RHDS server: `pki_ds_url=ldap://ldap.domain.com:389`

4.1.2. Differences between a root CA and a subordinate CA

For a root CA, the `caSigningCert` is self-signed in `pkispawn` phase 1.

- `pki_external=False`

For a subordinate CA, phase 1 completes the basic setup and creates the `caSigningCert` key and certificate request. Phase 2 imports the signed CA certificate chain and finishes the setup.

- `pki_external=True`
- Phase 1: `pki_external_step_two=False`
- Phase 2: `pki_external_step_two=True`

4.1.3. Ports

The defaults work for a single instance on the host. If you are using multiple instances on the same host, it is recommended to use high ports, for example in the 63000 range.

- `pki_security_domain_https_port=8443`
- `pki_http_port=8080`
- `pki_https_port=8443`
- `pki_ajp_port=8009`
- `pki_tomcat_server_port=8005`

4.1.4. Certificate distinguished names (DNs)

The six certificate DNs, especially the DN of the CA's own certificate, are important to an enterprise-class PKI.

Change each `*_subject_dn=` value to the DN required by your policy.

- `pki_ca_signing_subject_dn=cn=<CA Common Name>,OU=Group,OU=Division,O=Company,C=US`
- `pki_sslserver_subject_dn=cn=<server FQDN>,OU=Group,OU=Division,O=Company,C=US`
- `pki_subsystem_subject_dn=cn=<CA Common Name> Subsystem Certificate,OU=Group,OU=Division,O=Company,C=US`
- `pki_admin_subject_dn=cn=<CA Common Name> Agent Certificate,OU=Group,OU=Division,O=Company,C=US`
- `pki_audit_signing_subject_dn=cn=<CA Common Name> Audit Certificate,OU=Group,OU=Division,O=Company,C=US`
- `pki_ocsp_signing_subject_dn=cn=<CA Common Name> OCSP Certificate,OU=Group,OU=Division,O=Company,C=US`

4.1.5. Algorithms and key size

Five key pairs are created during installation. For policy reasons, the keys should match.

For all keys, change all `*_key_algorithm=`, `*_key_size=`, and `*_key_type=` parameters to match your key configuration.

- CA signing key (`pki_ca_signing_key_*`)
- Instance subsystem key (`pki_subsystem_key_*`)
- SSL/TLS web server key (`pki_sslserver_key_*`)
- Internal OCSP (`pki_ocsp_signing_key_*`)
- Instance audit signing key (`pki_audit_signing_key_*`)

Example for the CA's signing key using RSA2048 and SHA-256:

- `pki_ca_signing_key_algorithm=SHA256withRSA`
- `pki_ca_signing_key_size=2048`
- `pki_ca_signing_key_type=rsa`

Example for the CA's signing key using ECC (nistp256) and SHA-256:

- `pki_ca_signing_key_algorithm=SHA256withEC`
- `pki_ca_signing_key_size=nistp256`
- `pki_ca_signing_key_type=ecc`

Example for the CA's signing key using a post-quantum algorithm (ML-DSA):

- `pki_ca_signing_key_algorithm=ML-DSA-44`
- `pki_ca_signing_key_size=44`
- `pki_ca_signing_key_type=mldsa`

4.1.6. nShield HSM

Change all `*_token=` variables to match the name of your OCS or Softcard token.

- `pki_audit_signing_token=OCS1`
- `pki_sslserver_token=OCS1`
- `pki_subsystem_token=OCS1`
- `pki_token_name=OCS1`
- `pki_ca_signing_token=OCS1`
- `pki_ocsp_signing_token=OCS1`

4.1.7. Account passwords

Change all applicable `*_password=` variables from the defaults.

`pki_replication_password` is used only for cloning CAs.

- For the RHCS security domain, to join additional subsystems to the CA's security domain (`pki_security_domain_password`)
- For the RHCS instance's NSS database (`pki_server_database_password`)
- For the LDAP directory server (`pki_ds_password`)
- For the HSM token (`pki_token_password`)

The following default administrator passwords should match:

- Administrator credential, for example for logging in to pkiconsole (`pki_admin_password`)
- Default administrator credential in an NSS database (`pki_client_database_password`)
- Default administrator credential in a PKCS #12 file (`pki_client_pkcs12_password`)

4.1.8. Default agent credential

Load this PKCS #12 file into Firefox, or another web browser on any system, to access the agent web page and issue certificates or CRLs.

After the installation completes, this file is located in `/etc/pki/<instance>/agent_alias`.

<instance> is the `pki_instance_name` variable in the `pkispawn` configuration file.

4.1.9. Post-quantum limitation

The SSL server certificate (`pki_sslserver_*`) cannot use a post-quantum algorithm (ML-DSA). All other system certificates (CA signing, OCSP signing, audit signing, and subsystem) can use ML-DSA.

When the SSL server certificate is configured with ML-DSA:

```
pki_sslserver_key_algorithm=ML-DSA-44
pki_sslserver_key_size=44
pki_sslserver_key_type=mldsa
pki_sslserver_signing_algorithm=ML-DSA-44
```

`pkispawn` generates the keys and certificates and starts Tomcat, but fails when it checks the server status over HTTPS:

```
INFO: Waiting for PKI server to start (1s)
DEBUG: Starting new HTTPS connection (1): pki.domain.com:8443
ERROR: Exception: Server unreachable due to SSL error: [SSL: BAD_SIGNATURE] bad signature (_ssl.c:998)
  File "/usr/lib/python3.12/site-packages/pki/server/pkispawn.py", line 594, in main
    deployer.spawn()
  File "/usr/lib/python3.12/site-packages/pki/server/deployment/__init__.py", line 6330, in spawn
    scriptlet.spawn(self)
  File "/usr/lib/python3.12/site-packages/pki/server/deployment/scriptlets/configuration.py", line 247, in spawn
    instance.start(
  File "/usr/lib/python3.12/site-packages/pki/server/__init__.py", line 536, in start
    raise Exception('Server unreachable due to SSL error: %s' % reason) from e

Installation failed: Server unreachable due to SSL error: [SSL: BAD_SIGNATURE] bad signature (_ssl.c:998)
```

- **Cause**: The `pkispawn` client (Python 3.12 `ssl`, backed by the system OpenSSL) rejects the ML-DSA-44 signature that the server produces during the TLS handshake (signed through JSS/NSS and the nShield PKCS #11 library). ML-DSA works for certificate signatures, but ML-DSA TLS server authentication between these components does not work in this configuration. Most browsers also do not yet support ML-DSA TLS server certificates.
- **Workaround**: Use a classical algorithm for the SSL server key only. The server certificate is still signed by the ML-DSA CA. The `pkispawn.cfg` files use the following settings:

```
pki_sslserver_key_algorithm=SHA256withRSA
pki_sslserver_key_size=2048
pki_sslserver_key_type=rsa
```

- **Result**: `sslserver key=rsaEncryption sig=ML-DSA-44 OK`

Chapter 5. Additional resources and related products

5.1. nShield HSMs

5.2. nShield as a Service

5.3. Entrust products

5.4. nShield product documentation