Apache HTTP Server

nShield® HSM Integration Guide - PKCS #11

2025-06-10

© 2025 Entrust Corporation. All rights reserved.

Table of Contents

Chapter 1. Introduction

This guide describes how to integrate an nShield HSM with the Apache HTTP Server using `mod_ssl` to serve HTTPS websites. The integration process uses the Public-Key Cryptography Standards (PKCS #11) interface. The HSM stores the Apache Server's SSL private key within its FIPS 140 Level 3 validated hardware. After the integration is complete, you can deploy this integration in a Kubernetes environment.

Throughout this guide, the term HSM refers to nShield Solo and nShield Connect units.

1.1. Product configurations

Entrust has successfully tested nShield HSM integration with the Apache server in the following configurations:

Product	Version
Operating System	Red Hat Enterprise Linux 9 X86-64
Apache version	2.4.62-4
OpenSSL version	OpenSSL 3.2.2
OpenSSL PKCS #11 version	openssl-pkcs11-0.4.11-9

1.1.1. Supported nShield features

Entrust has successfully tested nShield HSM integration with the following features:

Feature	Support
Softcards	Yes
Module-only key	Yes
OCS cards	Yes
nSaaS	Not tested

1.1.2. Supported nShield hardware and software versions

Entrust has successfully tested with the following nShield hardware and software versions:

1.1.2.1. Connect XC

Security World Software	Firmware	Image	OCS	Softcard	Module	FIPS Level 3
13.6.11	12.72.3 (FIPS 140-2 certified)	13.6.7	✓	✓	✓	✓

1.1.2.2. nShield 5C

Security World Software	Firmware	Image	OCS	Softcard	Module	FIPS Level 3
13.6.11	13.4.5 (FIPS 140-3 certified)	13.6.7	✓	✓	✓	✓

1.2. Requirements

Ensure that you have supported versions of the nShield, Apache, and third-party products. See Product configurations.

Consult the security team in your organization for a suitable setting of the SE Linux policy to allow the web server read access to the files in `/opt/nfast`.

To perform the integration tasks, you must have:

- `root` access on the operating system.
- Access to `nfast` and `httpd` accounts.

Before starting the integration process, familiarize yourself with:

- The documentation for the HSM.
- The documentation and setup process for the Apache HTTP Server.

Before using the nShield software, you need to know:

- The number and quorum of Administrator Cards in the Administrator Card Set (ACS), and the policy for managing these cards.
- Whether the application keys are protected by the module, an Operator Card Set (OCS) or a Softcard with or without a pass phrase.
- The number and quorum of Operator Cards in the OCS, and the policy for managing these cards.
- Whether the Security World should be compliant with FIPS 140 Level 3.

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

For more information, refer to the *User Guide* and *Installation Guide* for the HSM.

1.3. More information

For more information about OS support, contact your Apache HTTP Server sales representative or Entrust nShield Support, https://nshieldsupport.entrust.com.

Access to the Entrust nShield Support Portal is available to customers under maintenance. To request an account, contact nshield.support@entrust.com.

Chapter 2. Procedures

2.1. Installing and configuring the Apache HTTP server

To install the Apache HTTP Server on your Red Hat server:

```
% sudo yum install -y openssl-pkcs11 httpd httpd-tools openssl-libs mod_ssl opensc
```

2.1.1. Open the firewall

If the firewall is active, this might prevent Apache from loading the library. To open the firewall:

```
% sudo firewall-cmd --zone=public --permanent --add-service=http
% sudo firewall-cmd --zone=public --permanent --add-service=https
% sudo firewall-cmd --reload
```

Consult the security team in your organization for suitable setting of the firewall.

2.1.2. Switch off SE Linux

If SE Linux is active, this might prevent Apache from loading the library. To switch it off:

```
% sudo setenforce 0
```

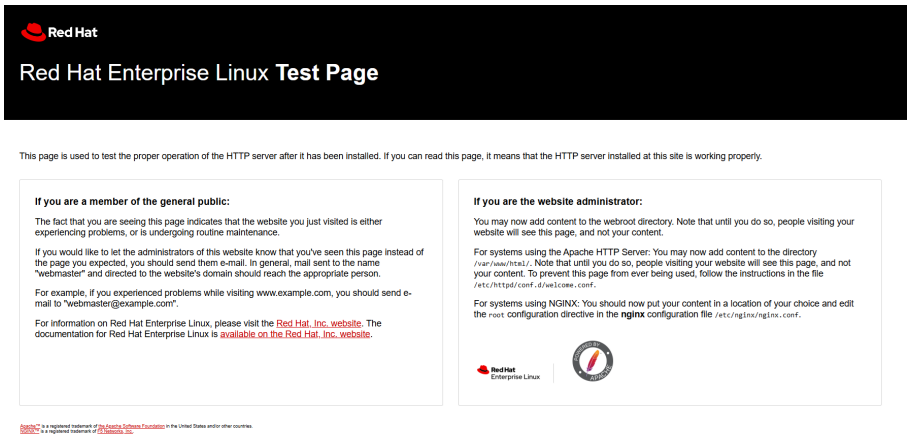
Consult the security team in your organization for a suitable setting of the SE Linux policy to allow the web server read access to the files in `/opt/nfast`.

2.1.3. Restart the httpd service

```
% sudo service httpd restart
```

2.1.4. Check if Apache is running

Open a browser and access the following URL: http://<YOUR_IP_ADDRESS>. You should see a page similar to this:



2.2. Install the HSM

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

Entrust recommends that you install the HSM before configuring the Security World software with your Apache HTTP Server.

2.3. Install the Security World software and create the Security World

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

1. On the computer that you want to make the Apache HTTP Server, install the latest version of the Security World Software as described in the *Installation Guide* for the HSM.

   Entrust recommends that you uninstall any existing nShield software before installing the new nShield software.

2. Create the Security World as described in the *User Guide*, creating the ACS and OCS that you require.

2.4. Set up the PKCS11 engine

To avoid problems associated with the Entrust supplied OpenSSL, which is used internally by `generatekey` to make certificates, ensure that `/opt/nfast/bin` is not at the front of your `$PATH`.

You can confirm that the right binary is being run with the following command:

```
% which openssl

/usr/bin/openssl
```

If this command returns anything inside `/opt/nfast`, check your `$PATH` variable.

2.4.1. Configuration

Find out where your OpenSSL configuration file is located:

```
% openssl version -d

OPENSSLDIR: "/etc/pki/tls"
```

The minimum configuration is similar to the following:

```
HOME = .

openssl_conf = openssl_def

[openssl_def]
engines = engine_section

[engine_section]
pkcs11 = pkcs11_section

[pkcs11_section]
engine_id = pkcs11
dynamic_path = /usr/lib64/engines-3/pkcs11.so #RHEL 9
MODULE_PATH = /opt/nfast/toolkits/pkcs11/libcknfast.so
init = 0
```

The following message can appear when creating certificates:

```
unable to find 'distinguished_name' in config
problems making Certificate Request
140493626791824:error:0E06D06C:configuration file routines:NCONF_get_string:no value:conf_lib.c:324:group=req
name=distinguished_name
```

If it does, you need to add the following to your OpenSSL configuration, adjusted to your organization's values:

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no

[req_distinguished_name]
C = US
ST = FL
```

```
L = Sunrise
O = Entrust
OU = nShield
CN = localhost

[v3_req]
subjectAltName = @alt_names
extendedKeyUsage = clientAuth, serverAuth

[alt_names]
DNS.1 = www.entrust.com
IP.1 = entrust.com
```

Make sure the server's hostname matches the CN in the certificate.

Create a file called `openssl.pkcs11.cnf` with the settings above, and save it where your OpenSSL configuration settings are located:

1. Create/edit the `/etc/pki/tls/openssl.pkcs11.cnf` file:

   ```
   % sudo vi /etc/pki/tls/openssl.pkcs11.cnf
   ```

2. Enter the settings above and save the file.

2.4.2. Set up /opt/nfast/cknfastrc

The following variables may need to be added to the `/opt/nfast/cknfastrc` file. They are referenced in this guide to address specific situations, and their use will depend on your current environment.

```
CKNFAST_DEBUG=10
CKNFAST_DEBUGFILE=/path/to/debug/file
CKNFAST_FAKE_ACCELERATOR_LOGIN=1
CKNFAST_LOADSHARING=1
```

If you omit `CKNFAST_DEBUGFILE`, log entries will be added to Apache's `error_log`.

Turn debug off in a production environment.

2.4.3. Test the configuration

You must now test the new configuration file using OpenSSL:

1. Exporting the `OPENSSL_CONF` environment variable:

   ```
   % export OPENSSL_CONF=/etc/pki/tls/openssl.pkcs11.cnf
   ```

2. Test the configuration:

```
% openssl engine -tt -c -v

There may be other output, but you should see this included:

(rdrand) Intel RDRAND engine
 [RAND]
     [ available ]
(dynamic) Dynamic engine loading support
     [ unavailable ]
     SO_PATH, NO_VCHECK, ID, LIST_ADD, DIR_LOAD, DIR_ADD, LOAD
(pkcs11) pkcs11 engine
 [RSA, rsaEncryption, id-ecPublicKey]
     [ available ]
     SO_PATH, MODULE_PATH, PIN, VERBOSE, QUIET, INIT_ARGS, FORCE_LOGIN,
     RE_ENUMERATE
```

2.4.3.1. Debugging notes

1. Security World permissions.

   The following message can appear:

   ```
   Unable to load module /opt/nfast/toolkits/pkcs11/libcknfast.so
   ```

   If it does, it indicates that there is no Security World. Make sure you create a Security world first.

2. Debugging variables.

   These variables can be used for debugging purpose. They can be set in `/opt/nfast/cknfastr` or as environment variables.

   ```
   CKNFAST_DEBUG=10
   CKNFAST_DEBUGFILE=/path
   ```

3. Missing PKCS #11 engine in the output.

   If you don't see the PKCS #11 engine in the output, check the `dynamic_path` line in the `openssl.pkcs11.cnf` configuration file. This may vary on other platforms and other operating system versions.

   ```
   dynamic_path = /usr/lib64/engines-3/pkcs11.so #RHEL 9
   ```

2.5. Configure the Apache HTTP Server to use the PKCS #11 engine

You need to update the Apache start-up file to tell it to use the new Open SSL configuration file, and to pass the necessary environment variables. These environment variables allow PKCS #11 engine to work.

1. Edit the file `/usr/lib/systemd/system/httpd.service` and add the environment variable under the “Service” section:

```
[Service]
Environment="OPENSSL_CONF=/etc/pki/tls/openssl.pkcs11.cnf"
```

2. Restart the daemon units:

```
% sudo systemctl daemon-reload
```

3. Restart the Apache service:

```
% sudo service httpd restart
```

4. Set the environment variable so that OpenSSL commands use the PKCS #11 engine:

```
% export OPENSSL_CONF=/etc/pki/tls/openssl.pkcs11.cnf
```

2.6. Test the PKCS #11 integration with the Apache HTTP Server and the HSM

This section describes the following scenarios that can be used by your organization according to the security guidelines that you follow:

- Module-only protection.
- Softcard protection.
- OCS protection.

A self-signed certificate is required for tests. In a production environment exposed to the internet, create the certificate request and sign it by the Trusted Certificate Authority. If using a FIPS Level 3 world file, have an OCS card connected to provide FIPS Authorization.

2.6.1. Module protection

1. Edit the `/opt/nfast/cknfastrc` file:

```
CKNFAST_FAKE_ACCELERATOR_LOGIN=1
```

2. Create a key:

```
% generatekey -b -g -m1 pkcs11 plainname=modulersa type=rsa protect=module size=2048

key generation parameters:
 operation    Operation to perform              generate
 application  Application                       pkcs11
 protect      Protected by                      module
 verify       Verify security of key            yes
 type         Key type                          rsa
 size         Key size                          2048
 pubexp       Public exponent for RSA key (hex)
 plainname    Key name                          modulersa
 nvram        Blob in NVRAM (needs ACS)         no
Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_uae95fb7af0294b94f742b22c62812fd0f18e0cf24
```

3. Get the certificate using this key:

```
% openssl req -engine pkcs11 -x509 -out modulersa.crt -days 365 -key
"pkcs11:token=accelerator;object=modulersa" -keyform engine -subj "/CN=modulersa"
```

4. Configure the Apache HTTP Server for SSL:
   a. Copy the `.crt` file:

```
% sudo cp modulersa.crt /etc/pki/tls/certs/.
```

   b. Edit `/etc/httpd/conf.d/ssl.conf` and change the following lines to use the new `.key` and `.crt` files:

```
SSLCertificateFile /etc/pki/tls/certs/modulersa.crt
SSLCertificateKeyFile "pkcs11:object=modulersa;token=accelerator"
SSLCryptoDevice pkcs11
```

   c. Restart the Apache service:

```
% sudo service httpd restart
```

5. Test the connections:

```
% openssl s_client -crlf -connect localhost:443 -CAfile modulersa.crt
```

6. Check the following messages and fields in the output:
    - CONNECTED(00000003)
    - depth
    - Certificate chain information
    - Server certificate information
    - Session-ID
    - Master-Key
    - TLS session ticket:
    - Verify return code: 0 (ok)

2.6.2. Set up Softcard protection

1. To expose Softcards, the `cknfast` library has to be in load sharing mode (`CKNFAST_LOADSHARING`).

   Edit the `/opt/nfast/cknfastrc` file, and add the following information before proceeding to set up Softcard protection:

   ```
   CKNFAST_LOADSHARING=1
   ```

2. Create a Softcard:

   ```
   % ppmk -n apachesoftcard
   ```

   Use "123456" as the passphrase for the Softcard.

3. Create a key:

   ```
   % generatekey -b -g -m1 pkcs11 plainname=softcardkey type=rsa protect=softcard recovery=no size=2048
   softcard=apachesoftcard

   key generation parameters:
    operation    Operation to perform              generate
    application  Application                       pkcs11
    protect      Protected by                      softcard
    softcard     Soft card to protect key          apachesoftcard
    recovery     Key recovery                      no
    verify       Verify security of key            yes
    type         Key type                          rsa
    size         Key size                          2048
    pubexp       Public exponent for RSA key (hex)
    plainname    Key name                          softcardkey
    nvram        Blob in NVRAM (needs ACS)         no
   Please enter the pass phrase for softcard `apachesoftcard':

   Please wait........
   Key successfully generated.
   ```

```
Path to key: /opt/nfast/kmdata/local/key_pkcs11_ucb87f22b0df8d3b72a2f4c654ae1d3b0973b93de8-
ddd20b997d276f3304e0011fc79971344c630b0f
```

4. Get the certificate using this key:

```
% openssl req -engine pkcs11 -x509 -out softcard.crt -days 365 -key
"pkcs11:model=;token=apachesoftcard;pin-value=123456;object=softcardkey" -keyform ENGINE -subj
"/CN=softcardkey"
```

   The following error can appear:

```
engine "pkcs11" set.
Specified object not found
PKCS11_get_private_key returned NULL
cannot load Private Key from engine
139939575797568:error:80067065:pkcs11 engine:ctx_load_privkey:object not found:eng_back.c:975:
139939575797568:error:26096080:engine routines:ENGINE_load_private_key:failed loading private
key:crypto/engine/eng_pkey.c:78:
```

   If it does, make sure you expose the Softcards as described in this section, and run the command again.

5. Configure the Apache HTTP Server for SSL:
   a. Copy the `.crt` file:

```
% sudo cp softcard.crt /etc/pki/tls/certs/.
```

   b. Edit `/etc/httpd/conf.d/ssl.conf` and change the following lines to use the new `.key` and `.crt` files:

```
SSLCertificateFile /etc/pki/tls/certs/softcard.crt
SSLCertificateKeyFile "pkcs11:object=softcardkey;token=apachesoftcard;type=private?pin-value=123456"
SSLCryptoDevice pkcs11
```

   c. Restart the Apache service:

```
% sudo service httpd restart
```

6. Test the connections:

```
% openssl s_client -crlf -connect localhost:443 -CAfile softcard.crt
```

7. Check the following messages and fields in the output:
   - CONNECTED(00000003)

- depth
- Certificate chain information
- Server certificate information
- Session-ID
- Master-Key
- TLS session ticket:
- Verify return code: 0 (ok)

2.6.3. Set up OCS protection

1. To expose OCS card, the `cknfast` library has to be in load sharing mode (`CKNFAST_LOADSHARING`).

   Edit the `/opt/nfast/cknfastrc` file, and add the following information before proceeding to set up OCS protection:

   ```
   CKNFAST_LOADSHARING=1
   ```

2. Create an OCS:

   ```
   % /opt/nfast/bin/createocs -m1 -s0 --persist -Q 1/1 -N apacheocs
   ```

   Use "123456" as the passphrase.

3. Create a key:

   ```
   % generatekey --cardset=apacheocs pkcs11 protect=token type=rsa size=2048 pubexp=65537 plainname=ocskey
   nvram=no recovery=yes

   slot: Slot to read cards from? (0-3) [0] > 0
   key generation parameters:
    operation    Operation to perform              generate
    application  Application                       pkcs11
    protect      Protected by                      token
    slot         Slot to read cards from           0
    recovery     Key recovery                      yes
    verify       Verify security of key            yes
    type         Key type                          rsa
    size         Key size                          2048
    pubexp       Public exponent for RSA key (hex) 65537
    plainname    Key name                          ocskey
    nvram        Blob in NVRAM (needs ACS)         no

   Loading `apacheocs':
    Module 1: 0 cards of 1 read
    Module 1 slot 0: `apacheocs' #1
    Module 1 slot 2: Admin Card #1
    Module 1 slot 3: empty
   ```

```
 Module 1 slot 0:- passphrase supplied - reading card
Card reading complete.

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_uc547fb435172da4280cc771eb3c2ad8b86ab06d0a-
8d6a4394b07fc70148ff9c1f9960d279bf4a1d6b
```

4. Get the certificate using this key:

```
% openssl req -engine pkcs11 -x509 -out ocskey.crt -days 365 -key
"pkcs11:token=apacheocs;object=ocskey;type=private?pin-value=123456" -keyform engine -subj "/CN=ocskey"
```

5. Configure the Apache HTTP Server for SSL:
   a. Copy the `.crt` file:

```
% sudo cp ocskey.crt /etc/pki/tls/certs/.
```

   b. Edit `/etc/httpd/conf.d/ssl.conf` and change the following lines to use the new `.key` and `.crt` files:

```
SSLCertificateFile /etc/pki/tls/certs/ocskey.crt
SSLCertificateKeyFile "pkcs11:object=ocskey;token=apacheocs;type=private?pin-value=123456"
SSLCryptoDevice pkcs11
```

   c. Restart the Apache service:

```
% sudo service httpd restart
```

6. Test the connections:

```
% openssl s_client -crlf -connect localhost:443 -CAfile ocskey.crt
```

7. Check the following messages and fields in the output:
   - CONNECTED(00000003)
   - depth
   - Certificate chain information
   - Server certificate information
   - Session-ID
   - Master-Key
   - TLS session ticket:
   - Verify return code: 0 (ok)

Chapter 3. Additional resources and related products

3.1. nShield Connect

3.2. nShield as a Service

3.3. Entrust products

3.4. nShield product documentation