NGINX: Integration Guide

THALES LUNA HSM AND DPOD LUNA CLOUD HSM

Document Information

Document Part Number	007-013662-001
Revision	E
Release Date	26 December 2022

Trademarks, Copyrights, and Third-Party Software

CONTENTS

Overview

This document guides you through the steps to integrate NGINX with Luna HSM and Luna Cloud HSM. NGINX is an open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. It is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. You can integrate NGINX with Luna HSM to generate 2048-bit RSA key pairs for SSL and protect the private keys within a FIPS 140-2 certified hardware security module. The benefits of integrating NGINX with Luna HSM include:

> Secure generation, storage, and protection of SSL keys on FIPS 140-2 level 3 validated hardware.

> Complete life cycle management of keys.

> Access to HSM audit trail*.

> Significant performance improvements by off-loading cryptographic operations from servers.

> Ability to use Cloud Services with confidence.

*Luna Cloud HSM services do not have access to the secure audit trail.

Certified Platforms

This integration is certified on the following platforms:

HSM Type	Operating System
Luna HSM	RHEL 8.x RHEL 7.x Ubuntu

NOTE: This integration is tested with Luna HSM clients in FIPS and HA Mode.

Luna HSM: Luna HSM appliances are purposefully designed to provide a balance of security, high performance, and usability that makes them an ideal choice for enterprise, financial, and government organizations. Luna HSMs physically and logically secure cryptographic keys and accelerate cryptographic processing. Luna HSM on premise offerings include the Luna Network HSM, Luna PCIe HSM, and Luna USB HSMs. Luna HSMs are also available for access as an offering from cloud service providers such as IBM cloud HSM and AWS cloud HSM classic.

HSM Type	Operating System
Luna Cloud HSM	RHEL 7.x Ubuntu

Luna Cloud HSM: Luna Cloud HSM provides on-demand HSM and Key Management services through a simple graphical user interface. With Luna Cloud HSM, security is simple, cost effective and easy to manage because there is no hardware to buy, deploy and maintain. As an Application Owner, you click and deploy services, generate usage reports and maintain just the services you need.

Prerequisites

Before you begin this integration, complete the following tasks:

Configure Luna HSM

If you are using Luna HSM:

1. Verify that the HSM is set up, initialized, provisioned, and ready for deployment. Refer to the Luna HSM documentation for more information.
2. Create a partition that will be later used by NGINX.
3. If using a Luna Network HSM, register a client for the system and assign the client to the partition to create an NTLS connection. Initialize the Crypto Officer and Crypto User roles for the registered partition.
4. Ensure that the partition is successfully registered and configured. The command to view the registered partitions is:

```
# /usr/safenet/lunaclient/bin/lunacm
lunacm (64-bit) v10.4.0-417. Copyright (c) 2021 SafeNet. All rights reserved.

       Available HSMs:

       Slot Id ->              0
       Label ->                TPA01
       Serial Number ->        1312109862206
       Model ->                LunaSA 7.7.1
       Firmware Version ->     7.7.1
       Bootloader Version ->   1.1.2
       Configuration ->        Luna User Partition With SO (PW) Key Export
                               With Cloning Mode
       Slot Description ->     Net Token Slot
       FM HW Status ->         Non-FM

       Current Slot Id: 0
```

5. For PED-authenticated HSM, enable partition policies 22 and 23 to allow activation and auto-activation.

> **NOTE:** Refer to Luna HSM documentation for detailed steps to create NTLS connection and initialize the partitions and various user roles.

Configure Luna HSM HA (High-Availability)

Refer to Luna HSM documentation for HA steps and details regarding configuring and setting up two or more HSM appliances on Windows and UNIX systems. You must enable the HAOnly setting in HA for failover to work so that if primary stop functioning for some reason, all calls automatically routed to secondary till primary start functioning again.

Use Luna HSM in FIPS mode

Under FIPS 186-3/4, the RSA methods permitted for generating keys are 186-3 with primes and 186-3 with aux primes. This means that RSA PKCS and X9.31 key generation is no longer approved for operation in a FIPS-compliant HSM. If you are using the Luna HSM in FIPS mode, you have to make the following change in configuration file:

```
Misc = {
RSAKeyGenMechRemap = 1;
}
```

The above setting redirects the older calling mechanism to a new approved mechanism when Luna HSM is in FIPS mode.

> **NOTE:** The above setting is not required for the Universal Client. This setting is applicable only for Luna Clients 7.x.

Control user access to the HSM

By default, only the root user has access to the HSM. You can specify a set of non-root users that are permitted to access the HSM by adding them to the hsmusers group. The client software installation automatically creates the hsmusers group. The hsmusers group is retained when you uninstall the client software, allowing you to upgrade the software while retaining your hsmusers group configuration.

Add a user to hsmusers group

To allow non-root users or applications access to the HSM, assign the user to the hsmusers group. The users you assign to the hsmusers group must exist on the client workstation. To add a user to hsmusers group:

1. Ensure that you have sudo privileges on the client workstation.
2. Add a user to the hsmusers group.

   ```
   sudo gpasswd --add <username> hsmusers
   ```

   Where <username> is the name of the user you want to add to the hsmusers group.

Remove a user from hsmusers group

To remove a user from hsmusers group:

1. Ensure that you have sudo privileges on the client workstation.
2. Remove a user from the hsmusers group.

   ```
   sudo gpasswd -d <username> hsmusers
   ```

   Where <username> is the name of the user you want to remove from the hsmusers group. You must log in again to see the change.

   > **NOTE:** The user you delete will continue to have access to the HSM until you reboot the client workstation.

Configure Luna Cloud HSM service

You can configure Luna Cloud HSM Service in the following ways:

> Standalone Cloud HSM service using minimum client package

> Standalone Cloud HSM service using full Luna client package

> Luna HSM and Luna Cloud HSM service in hybrid mode

NOTE: Luna Client v10.x or higher is required for configuring Luna HSM device and Luna Cloud HSM service in hybrid mode.

Standalone Cloud HSM service using minimum client package

To configure Luna Cloud HSM service using minimum client package:

1. Transfer the downloaded .zip file to your Client workstation using pscp, scp, or other secure means.
2. Extract the .zip file into a directory on your client workstation.
3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the client install directory.

```
[Linux]

cvclient-min.tar

# tar -xvf cvclient-min.tar
```

4. Run the setenv script to create a new configuration file containing information required by the Luna Cloud HSM service.

```
[Linux]

Source the setenv script.

# source ./setenv
```

5. Run the LunaCM utility and verify the Cloud HSM service is listed.

Standalone Cloud HSM service using full Luna client package

To configure Luna Cloud HSM service using full Luna client package:

1. Transfer the downloaded .zip file to your Client workstation using pscp, scp, or other secure means.
2. Extract the .zip file to a directory on your client workstation.
3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the client install directory.

```
[Linux]

cvclient-min.tar

# tar -xvf cvclient-min.tar
```

4. Run the setenv script to create a new configuration file containing information required by the Luna Cloud HSM service.

```
[Linux]

Source the setenv script.

# source ./setenv
```

5. Copy the server and partition certificates from the Cloud HSM service client directory to the Luna client certificates directory:

   Cloud HSM Certificates:

   `server-certificate.pem`

   `partition-ca-certificate.pem`

   `partition-certificate.pem`

   LunaClient Certificate Directory:

   `[Linux default location for Luna Client]`

   `/usr/safenet/lunaclient/cert/`

   > **NOTE:** Skip this step for Luna Client v10.2 or higher.

6. Open the configuration file from the Cloud HSM service client directory and copy the XTC and REST section.

   `[Linux]`

   `Chrystoki.conf`

7. Edit the Luna Client configuration file and add the XTC and REST sections copied from Cloud HSM service client configuration file.

8. Change server and partition certificates path from step 5 in XTC and REST sections. Do not change any other entries provided in these sections.

   [XTC]

```
. . .
PartitionCAPath=<LunaClient_cert_directory>\partition-ca-certificate.pem
PartitionCertPath00=<LunaClient_cert_directory>\partition-certificate.pem
. . .

[REST]
. . .
SSLClientSideVerifyFile=<LunaClient_cert_directory>\server-certificate.pem
. . .
```

   > **NOTE:** Skip this step for Luna Client v10.2 or higher.

9. Edit the following entry from the Misc section and update the correct path for the plugins directory:

```
Misc]
PluginModuleDir=<LunaClient_plugins_directory>
```

   `[Linux Default]`

```
/usr/safenet/lunaclient/plugins/
```

10. Save the configuration file. If you wish, you can now safely delete the extracted Cloud HSM service client directory.

11. Reset the ChrystokiConfigurationPath environment variable and point back to the location of the Luna Client configuration file.

    Linux

    Either open a new shell session, or export the environment variable for the current session pointing to the location of the Chrystoki.conf file:

    ```
    # export ChrystokiConfigurationPath=/etc/
    ```

12. Run the LunaCM utility and verify that the Cloud HSM service is listed. In the hybrid mode, both Luna and Cloud HSM service will be listed.

    > **NOTE:** Follow the Luna Cloud HSM documentation for detailed steps for creating service, client, and initializing various user roles.

Luna HSM and Luna Cloud HSM service in hybrid mode

To configure Luna HSM and Luna Cloud HSM service in hybrid mode, follow the steps mentioned under the Standalone Cloud HSM service using full Luna client package section above.

> **NOTE:** Luna Client v10.x or higher is required for configuring Luna HSM device and Luna Cloud HSM service in hybrid mode.

To use Luna Cloud HSM Service in FIPS mode

Cloud HSM service operates in both FIPS and non-FIPS mode. If your organization requires non-FIPS algorithms for your operations, enable the Allow non-FIPS approved algorithms check box when configuring your Cloud HSM service. The FIPS mode is enabled by default. Refer to the Mechanism List in the SDK Reference Guide for more information about available FIPS and non-FIPS algorithms.

Set up NGINX

NGINX server must be installed on the target machines to enable the integration process. For a detailed installation procedure, refer to the NGINX documentation.

> **NOTE:** If you are using HSM in FIPS mode, NGINX must be compiled and installed with OpenSSL in FIPS mode.

Integrating Luna HSM with NGINX

Integration of Luna HSM with NGINX involves two use cases:

> Integrate Luna HSM with NGINX by generating new SSL keys

> Integrate Luna HSM with NGINX by migrating existing SSL keys

Integrate NGINX with Luna HSM by generating new SSL keys

To integrate NGINX with Luna HSM by generating new SSL keys, complete the following tasks:

> Configure OpenSSL to enable GemEngine

> Configure SSL for NGINX using OpenSSL

Configure OpenSSL to enable GemEngine

You can either configure OpenSSL that is installed by default in the system or you can install another version and configure it. To configure GemEngine for OpenSSL:

1. Log on to the NGINX server as a root user or as any other user with administrative privileges.
2. Download the OpenSSL toolkit from the Thales support portal, extract it on your system, and go to the directory where Gem Engine is extracted to locate the `gembuild` utility.

   For example:

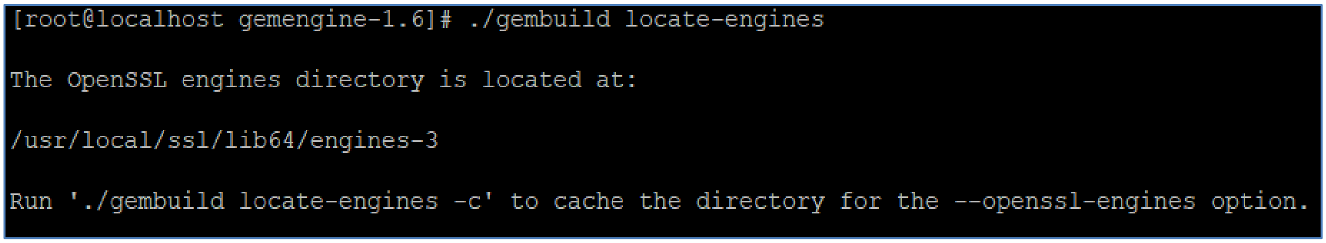
   ```
   # cd /home/gemengine-x.x
   ```

   Here, x.x is the Gem Engine version.

   > **NOTE:** NGINX uses OpenSSL for SSL/TLS support. OpenSSL includes a component called ENGINE to store keys on hardware devices. Thales provides the Gem Engine toolkit having support of ENGINE that is used to communicate with the Luna HSM. The Gem Engine toolkit can be download from the Thales Support Portal. It is recommended that you should familiarize yourself with OpenSSL. Refer to the appropriate documents for OpenSSL commands at http://www.openssl.org/docs/.

3. Locate the OpenSSL engines directory using the gembuild command.

   ```
   # ./gembuild locate-engines
   ```

   ```
   [root@localhost gemengine-1.6]# ./gembuild locate-engines

   The OpenSSL engines directory is located at:

   /usr/local/ssl/lib64/engines-3

   Run './gembuild locate-engines -c' to cache the directory for the --openssl-engines option.
   ```

4. Copy the **libgem.so** or **gem.so** to the engines directory displayed in the previous command, depending on the OpenSSL version.

   For example:

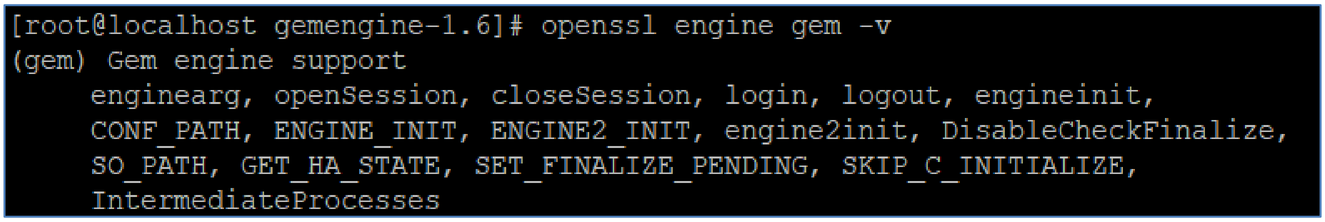
   ```
   # cp builds/linux/rhel/64/x.x.x/gem.so <OpenSSL engines directory path>
   ```

   Here, x.x.x is your OpenSSL version

   > **NOTE:** You can also build and install Gem Engine, SAUTIL, and OpenSSL using the Gem Engine Toolkit downloaded from Thales Portal. Refer to the README files provided with the Toolkit for detailed instructions.

5. Verify that Gem Engine is present and supported by OpenSSL.

   ```
   # openssl engine gem -v
   ```

   ```
   [root@localhost gemengine-1.6]# openssl engine gem -v
   (gem) Gem engine support
        enginearg, openSession, closeSession, login, logout, engineinit,
        CONF_PATH, ENGINE_INIT, ENGINE2_INIT, engine2init, DisableCheckFinalize,
        SO_PATH, GET_HA_STATE, SET_FINALIZE_PENDING, SKIP_C_INITIALIZE,
        IntermediateProcesses
   ```

6. Create a file containing partition CO password needed by Gem Engine to log on to the Luna partition.

```
# echo CO_Password > /tmp/passfile
```

7. Add support for Gem Engine in the /etc/Chrystoki.conf file. Update the Chrystoki.conf file as follows:

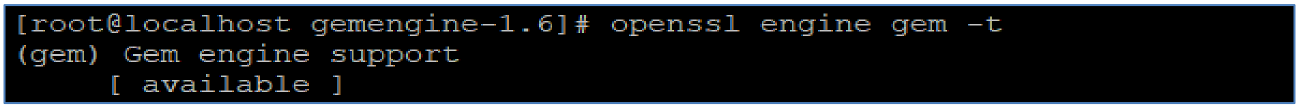
```
GemEngine = {
   LibPath = /usr/safenet/lunaclient/lib/libCryptoki2.so;
   LibPath64 = /usr/safenet/lunaclient/lib/libCryptoki2_64.so;
   EnableDsaGenKeyPair = 1;
   EnableRsaGenKeyPair = 1;
   DisablePublicCrypto = 1;
   EnableRsaSignVerify = 1;
   EnableLoadPubKey = 1;
   EnableLoadPrivKey = 1;
   DisableCheckFinalize = 1;
   DisableEcdsa = 1;
   DisableDsa = 0;
   DisableRand = 0;
   EngineInit = "<myTokenLabel>":0:0:passfile=</path/to/my/passfile>;
   EnableLoginInit = 1;
}
```

Here, `<myTokenLabel>` is the partition label and `</path/to/my/passfile>` is the path to file containing the partition CO pin.

> **NOTE:** If you do not want to save the partition's CO password in a file, several other methods are available to enable login via Gem Engine. Refer the README files provided with the Gem Engine Toolkit for detail instructions.

8. Verify that OpenSSL is configured successfully to start using Luna HSM with GemEngine.

```
# openssl engine gem -t
```



```
[root@localhost gemengine-1.6]# openssl engine gem -t
(gem) Gem engine support
     [ available ]
```

This completes the OpenSSL configuration for Gem Engine support.

Configure SSL for NGINX using OpenSSL

NGNIX server utilizes OpenSSL generated SSL keys and certificates for SSL communication. You need to generate certificate and keys using OpenSSL that leverages Gem Engine to generate key on Luna HSM. After key generation, update the NGINX configuration file to start SSL communication.

- Generate certificates
- Update NGINX to start SSL

Generate certificates

To configure SSL, you need to generate the certificate that can be either self-signed or signed by a renowned CA. In both cases, the certificate private key will be secured in Luna HSM. The steps to generate CA signed certificate and self-signed certificate are as follows:

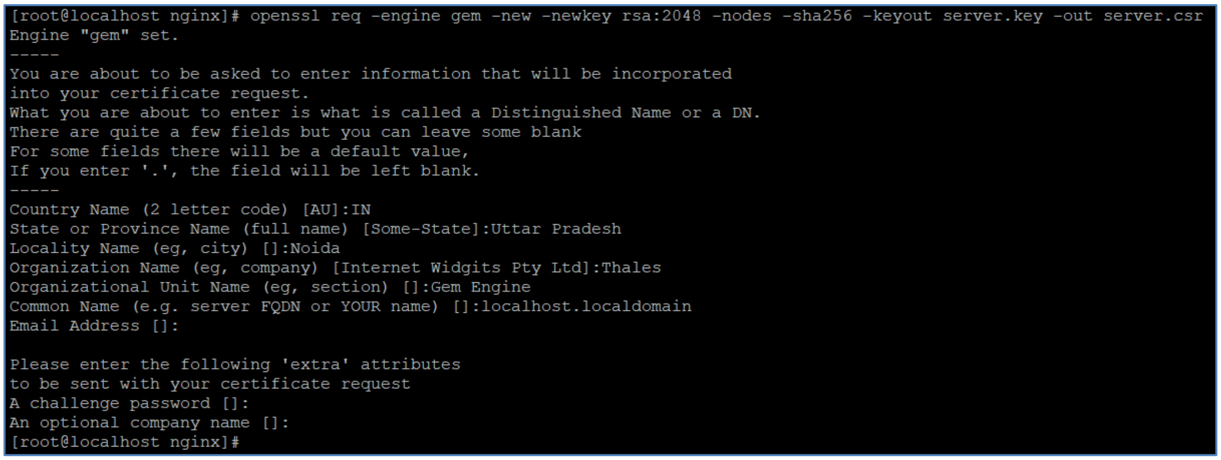
> **NOTE:** It is recommended to use the CA signed certificate in production environment. Self-Signed certificate is suitable for test environment only.

CA-signed SSL certificate

To generate the CA-signed SSL certificate:

1. Execute the command below to generate the keys on Luna HSM and save the certificate request and key reference.

```
# openssl req -engine gem -new -newkey rsa:2048 -nodes -sha256 -keyout
server.key -out server.csr
```



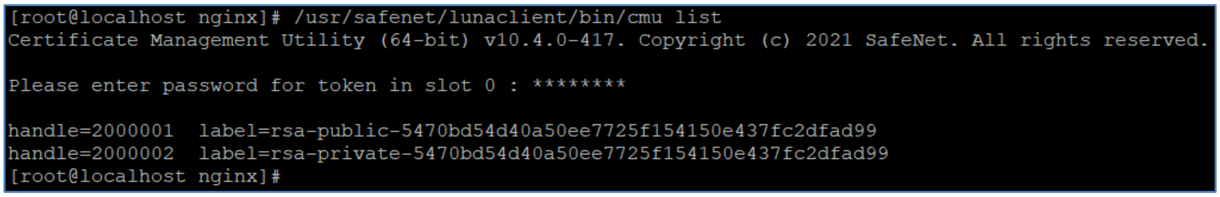
```
[root@localhost nginx]# openssl req -engine gem -new -newkey rsa:2048 -nodes -sha256 -keyout server.key -out server.csr
Engine "gem" set.
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:Uttar Pradesh
Locality Name (eg, city) []:Noida
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Thales
Organizational Unit Name (eg, section) []:Gem Engine
Common Name (e.g. server FQDN or YOUR name) []:localhost.localdomain
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[root@localhost nginx]#
```

The public key and private key will be generated on the HSM and the private key reference generated on the HSM will be saved in the **server.key** file. You'll be requiring this later. The Certificate Signing Request (CSR) will be saved in the **server.csr** file that needs to be submitted to the CA for obtaining a CA-signed certificate.

2. Run the **cmu** list to verify the generated key pair on Luna HSM.

```
# /usr/safenet/lunaclient/bin/cmu list
```

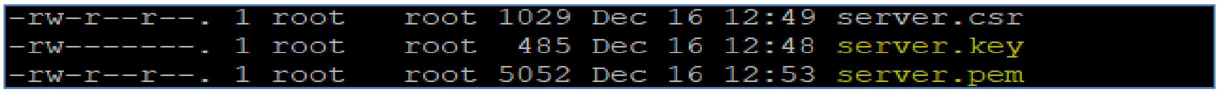


```
[root@localhost nginx]# /usr/safenet/lunaclient/bin/cmu list
Certificate Management Utility (64-bit) v10.4.0-417. Copyright (c) 2021 SafeNet. All rights reserved.

Please enter password for token in slot 0 : ********

handle=2000001  label=rsa-public-5470bd54d40a50ee7725f154150e437fc2dfad99
handle=2000002  label=rsa-private-5470bd54d40a50ee7725f154150e437fc2dfad99
[root@localhost nginx]#
```

3. Submit the CSR file (server.csr) to a CA such as VeriSign or Entrust. The CA authenticates the request and returns a signed certificate or a certificate chain. Save the CA-signed certificate in the system directory. Provide your key reference (server.key) and CA signed certificate (server.pem) in the NGINX configuration.



```
-rw-r--r--. 1 root   root 1029 Dec 16 12:49 server.csr
-rw-------. 1 root   root  485 Dec 16 12:48 server.key
-rw-r--r--. 1 root   root 5052 Dec 16 12:53 server.pem
```

Self-signed SSL certificate

To generate self-signed SSL certificate:

1. Execute the command below to generate the keys on Luna HSM and save the key reference.

   ```
   # openssl genrsa -engine gem -out server.key 2048
   ```

   The server.key is the key reference to Private Key Generated on HSM. You will require it later.

2. To generate a self-signed certificate that can be be used for test purpose, execute the following command.

   ```
   # openssl req -new -engine gem -x509 -key server.key -sha256 -out server.pem
   ```

   Here, server.pem is the self-signed certificate in PEM format.

Update NGINX to start SSL

For updating NGINX to use Luna HSM generated SSL keys:

1. Open the NGINX configuration file <NGINX installation directory>/nginx.conf file and update the file as follows to enable the SSL support at the end of http section in nginx.conf.

```
    server {
        listen       443 ssl;
        server_name  <Server Hostname or IP Address>;

        ssl_certificate      <Path to the certificate.pem file>;
        ssl_certificate_key  <Path to the private key file >;

        ssl_session_cache    shared:SSL:1m;
        ssl_session_timeout  5m;

        ssl_protocols   TLSv1.2 TLSv1.3;
        ssl_ciphers  HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers  on;

        location / {
            root   html;
            index  index.html index.htm;
        }
    }
```

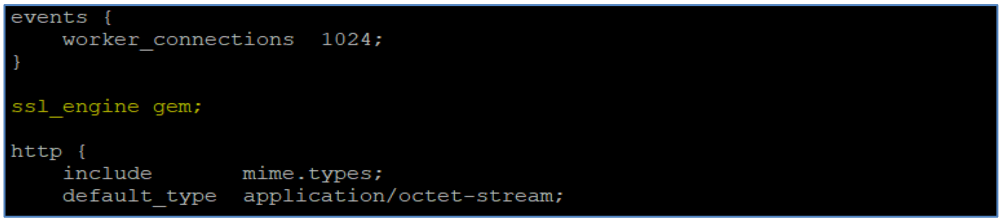
Replace the server_name, ssl_certificate and ssl_certificate_key values with the actual values in your environment. Here:

- ssl_certificate is the self-signed or CA signed certificate.
- ssl_certificate_key is the location of reference to private key generated on the HSM in PEM format.

> **NOTE:** TLSv1.3 support is available in OpenSSL v1.1.1 onwards. Older OpenSSL versions do not supports TLSv1.3.

2. Open the NGINX configuration file <NGINX installation directory>/nginx.conf file and update the nginx.conf file as follows to enable the Gem Engine support before the beginning of http section.

   ```
   ssl_engine gem;
   ```

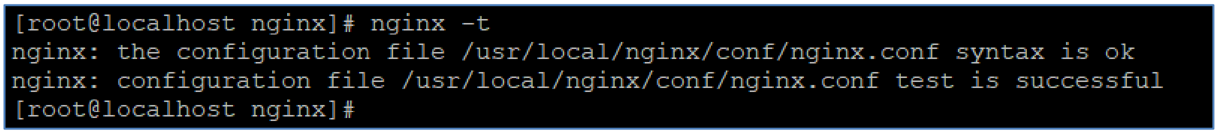
   ```
   events {
       worker_connections  1024;
   }

   ssl_engine gem;

   http {
       include       mime.types;
       default_type  application/octet-stream;
   ```

3. Run the following command under the <NGINX installation directory>/bin/ to verify that there is no error due to the configuration changes in the nginx.conf file.

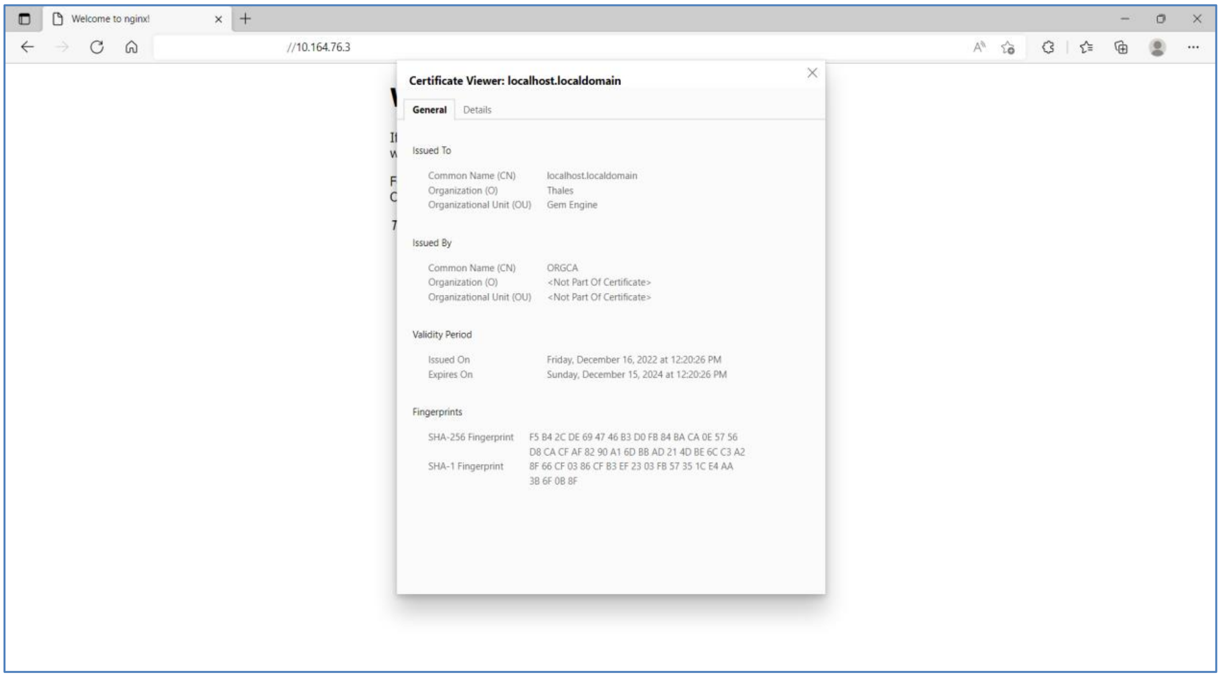
   ```
   # ./nginx -t
   ```

   ```
   [root@localhost nginx]# nginx -t
   nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
   nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
   [root@localhost nginx]#
   ```

4. Restart the NGINX server.

5. Open any web browser and access the NGINX server. Verify the certificate.

   Certificate Viewer: localhost.localdomain

   Issued To

Common Name (CN)	localhost.localdomain
Organization (O)	Thales
Organizational Unit (OU)	Gem Engine

   Issued By

Common Name (CN)	ORGCA
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

   Validity Period

Issued On	Friday, December 16, 2022 at 12:20:26 PM
Expires On	Sunday, December 15, 2024 at 12:20:26 PM

   Fingerprints

SHA-256 Fingerprint	F5 B4 2C DE 69 47 46 B3 D0 FB 84 BA CA 0E 57 56 D8 CA CF AF 82 90 A1 6D BB AD 21 4D BE 6C C3 A2
SHA-1 Fingerprint	8F 66 CF 03 86 CF B3 EF 23 03 FB 57 35 1C E4 AA 3B 6F 0B 8F

This completes the NGINX integration with Luna HSM. The SSL private key of NGINX server is now safe and secured on Luna HSM partition. The SSL page will be displayed only if the HSM partition is accessible and available to the NGINX Server.

Integrate NGINX with Luna HSM by migrating existing SSL keys

It is assumed that the NGINX server is already configured and running on SSL, wherein SSL certificate and keys are generated by OpenSSL and saved in the system directory.

To integrate NGINX with Luna HSM by migrating the existing SSL keys:

1. Configure OpenSSL to use Gem Engine by executing the steps mentioned in the Configure OpenSSL to enable GemEngine section.
2. Locate the directory where the SSL private key and certificate are saved.
3. Extract the Certificate Public key using the command below.

   ```
   # openssl rsa -in server.key -pubout -out pubkey.pem
   ```

4. Extract the Private Key in PKCS#8 format using the below command.

   ```
   # openssl pkcs8 -in server.key -topk8 -nocrypt -out privatekey.pem
   ```

5. Import the public key and private key to the Luna HSM by using the CMU utility provided with Luna Client.

   For Public Key:

   ```
   # /usr/safenet/lunaclient/bin/cmu import -inputFile pubkey.pem -label
   nginx_public_key -pubkey=rsa
   ```

   For Private Key:

   ```
   # /usr/safenet/lunaclient/bin/cmu importkey -PKCS8 -in privatekey.pem -keyalg
   RSA
   ```

6. Verify that the keys are generated on Luna HSM partition and note the private key handle.

   ```
   # /usr/safenet/lunaclient/bin/cmu list

   Certificate Management Utility (64-bit) v10.1.0-32. Copyright (c) 2019 SafeNet.
   All rights reserved.

   Please enter password for token in slot 0 : ********

   handle=37       label=CMU Unwrapped RSA Private Key

   handle=36       label=nginx_public_key
   ```

7. Use the following command to set a label to easily recognize NGINX SSL private key:

   ```
   # /usr/safenet/lunaclient/bin/cmu setattribute -handle=37 -
   label=nginx_private_key
   ```

8. Verify that the private key label is set and matches the label of public key.

   ```
   # /usr/safenet/lunaclient/bin/cmu list

   Certificate Management Utility (64-bit) v10.3.0-140. Copyright (c) 2020
   SafeNet. All rights reserved.

   Please enter password for token in slot 0 : ********

   handle=37       label=nginx_private_key

   handle=36       label=nginx_public_key
   ```

9. Copy the SAUTIL utility provided with OpenSSL toolkit to create the Private Key reference of the key imported on Luna HSM partition.

   ```
   # cp /home/gemengine-1.2/builds/linux/rhel/64/1.0.2/sautil /usr/bin/
   ```

10. Run the **sautil** utility to create Private Key Reference to actual private key imported in Luna HSM.

    ```
    # sautil -v -s 0 -i 0:0 -a 0:RSA -f HSMKey_ref.pem -o -q –c
    ```

    Provide the HSM partition CO password and key handle when prompted. After the successful execution of `sautil` command, HSMKey_ref.pem will be generated and needs to be specified in the SSL configuration in nginx.conf file.

11. Remove the Private Key generated by OpenSSL that was used before importing the key in to Luna HSM along with the PKCS#8 format key generated in step 4.

    ```
    # rm -rf /usr/localnginx/server.key /usr/local/nginx/privatekey.pem
    ```

12. Edit the nginx.conf file and update the ssl_certificate_key location with the HSM Key Reference generated in step 10.

    ```
    server {
        listen       443 ssl;
        server_name  <Server Hostname or IP Address>;

        ssl_certificate      /usr/local/nginx/server.pem;
        ssl_certificate_key  /usr/local/nginx/HSMKey_ref.pem;

        ssl_session_cache    shared:SSL:1m;
        ssl_session_timeout  5m;

        ssl_protocols   TLSv1.2 TLSv1.3;
        ssl_ciphers  HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers  on;

        location / {
            root   html;
            index  index.html index.htm;
        }
    }
    ```

    > **NOTE:** TLSv1.3 support is available in OpenSSL v1.1.1 onwards. Older OpenSSL versions do not support TLSv1.3.

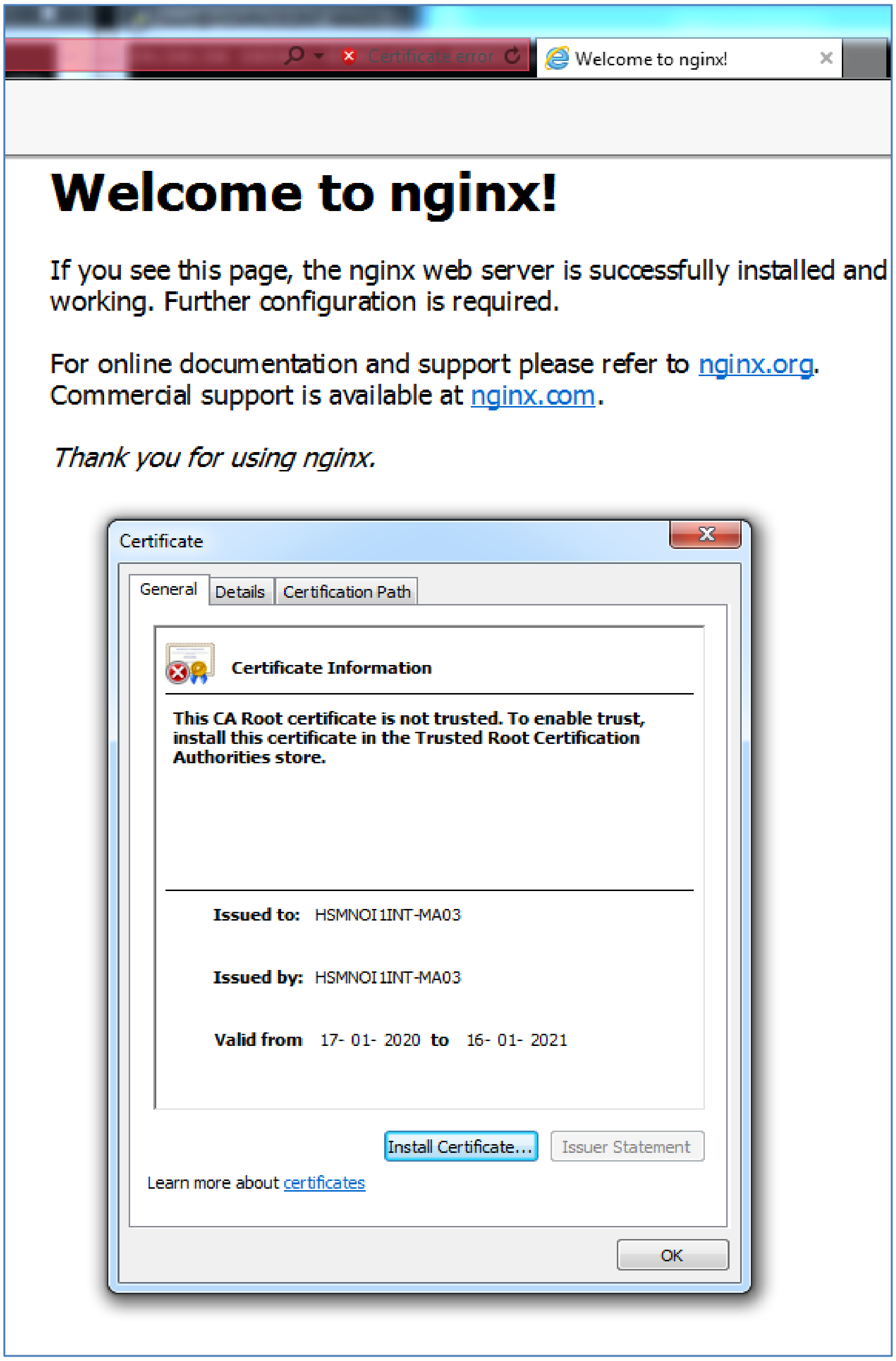
13. Replace the server_name, ssl_certificate, and ssl_certificate_key value with the actual values in your environment. Here, ssl_certificate is the self-signed or CA-signed certificate in the PEM format and ssl_certificate_key is the location of reference key pointing to private key imported on Luna HSM.

14. Open the NGINX configuration file <NGINX installation directory>/nginx.conf and update it as follows to enable the Gem Engine support before the beginning of http section.

    ```
    ssl_engine gem;
    ```

15. Restart the NGINX server.

16. Open any browser and access the NGINX server. Verify that the server is accessible and is using the private key migrated to the HSM.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: HSMNOI1INT-MA03

Issued by: HSMNOI1INT-MA03

Valid from 17- 01- 2020 **to** 16- 01- 2021

This completes the migration of NGINX SSL keys to Luna HSM partition.

Contacting Customer Support

If you encounter a problem during this integration, refer to the documentation. If you cannot resolve the issue, contact your supplier or Thales Customer Support. Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is a database where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable repository of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

> **NOTE:** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

Telephone Support

If you have an urgent problem, or cannot access the Customer Support Portal, you can contact Thales Customer Support by telephone at +1 410-931-7520. Additional local telephone support numbers are listed on the support portal.