THALES

Apache Tomcat: Integration Guide

THALES LUNA HSM

Document Information

Document Part Number	007-000637-001
Revision	D
Release Date	23 August 2023

Trademarks, Copyrights, and Third-Party Software

CONTENTS

Overview

This document provides you the necessary information for installing, configuring, and integrating Apache Tomcat with Thales Luna HSMs. The integration between Luna HSMs and Apache Tomcat leverages Java JCE/JCA interface to generate the SSL keys on Luna HSMs. You can also use native library to enable the APR connector that uses OpenSSL for its cryptographic operations. Luna HSMs integrate with Apache Tomcat to generate 2048-bit RSA key pairs for SSL encryption and provide security by protecting the private keys and certificates within a FIPS 140-2 certified hardware security module.

The benefits of using Luna HSMs to generate the SSL keys for Apache Tomcat include:

- Ensuring secure key generation, storage, and protection through FIPS 140-2 level 3 validated hardware.
- Providing full life cycle management of the keys.
- Maintaining an audit trail through HSM.
- Achieving significant performance enhancements by offloading cryptographic operations from application servers.

Supported Platforms

This integration is certified with Luna HSM on following platforms:

Apache Tomcat	Java	Platforms
Apache Tomcat 9.0.79	Open JDK 11	Red Hat Enterprise Linux 8
Apache Tomcat 10.0.6 (With Native Library)	Open JDK 11	Red Hat Enterprise Linux 8 and OpenSSL 1.1.1
Apache Tomcat 9.0.20 (With Native Library)	Open JDK 8	Solaris 11 and OpenSSL 1.1.1
Apache Tomcat 8.5.57	Open JDK 11	Red Hat Enterprise Linux 7
Apache Tomcat 9.0.31	Open JDK 8	Red Hat Enterprise Linux 7
Apache Tomcat 8.5.51	Oracle JDK 8	Windows Server 2016 Datacenter
Apache Tomcat 8.5.40	Open JDK 8	Red Hat Enterprise Linux 7
Apache Tomcat 8.5.40	Oracle JDK 8	Windows Server 2016 Datacenter

NOTE: For JDK 11, Luna Client 10.3.0 or above is required.

> Open JDK 11 is supported for Luna Client 10.2.0 also but with patch: **DOW0005918**. You can download this patch from Thales Customer Support portal if using Luna Client 10.2.0 with JDK 11. Lower versions of Luna Client don't support JDK 11.

Luna HSMs: 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. The Luna HSM on premise offerings include the Luna Network HSM, PCIe HSM, and Luna USB HSMs.

Prerequisites

Before you proceed with the integration, complete the following tasks:

Configure Luna HSM

Luna HSMs provides strong physical protection of secure assets, including keys, and should be considered a best practice when building systems based on Apache Tomcat.

To configure the Luna HSM:

1. Ensure that the HSM is set up, initialized, provisioned and ready for deployment. Refer to the HSM product documentation for help.
2. Create a partition that you'll be using for Apache Tomcat.
3. Create and exchange certificate between the Luna Network HSM and Client system. Register client and assign partition to create an NTLS connection. Initialize Crypto Officer and Crypto User roles for the registered partition.
4. Ensure that the partition is successfully registered and configured.

   The command to see 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:** Follow the Luna Network Luna HSM documentation for detailed steps for creating NTLS connection, initializing the partitions, and various user roles.

Controlling 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.

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

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 HSM HA (High-Availability)

Please refer to the 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 starts functioning again.

> **NOTE:** This integration is tested in both HA and FIPS mode.

Install Java Development Kit

Ensure that you have the Java Development Kit (JDK) installed. Apache Tomcat relies on Java, it's crucial to have the right JDK version. Make sure JAVA_HOME points to your JDK installation directory.

Set up Gem Engine toolkit

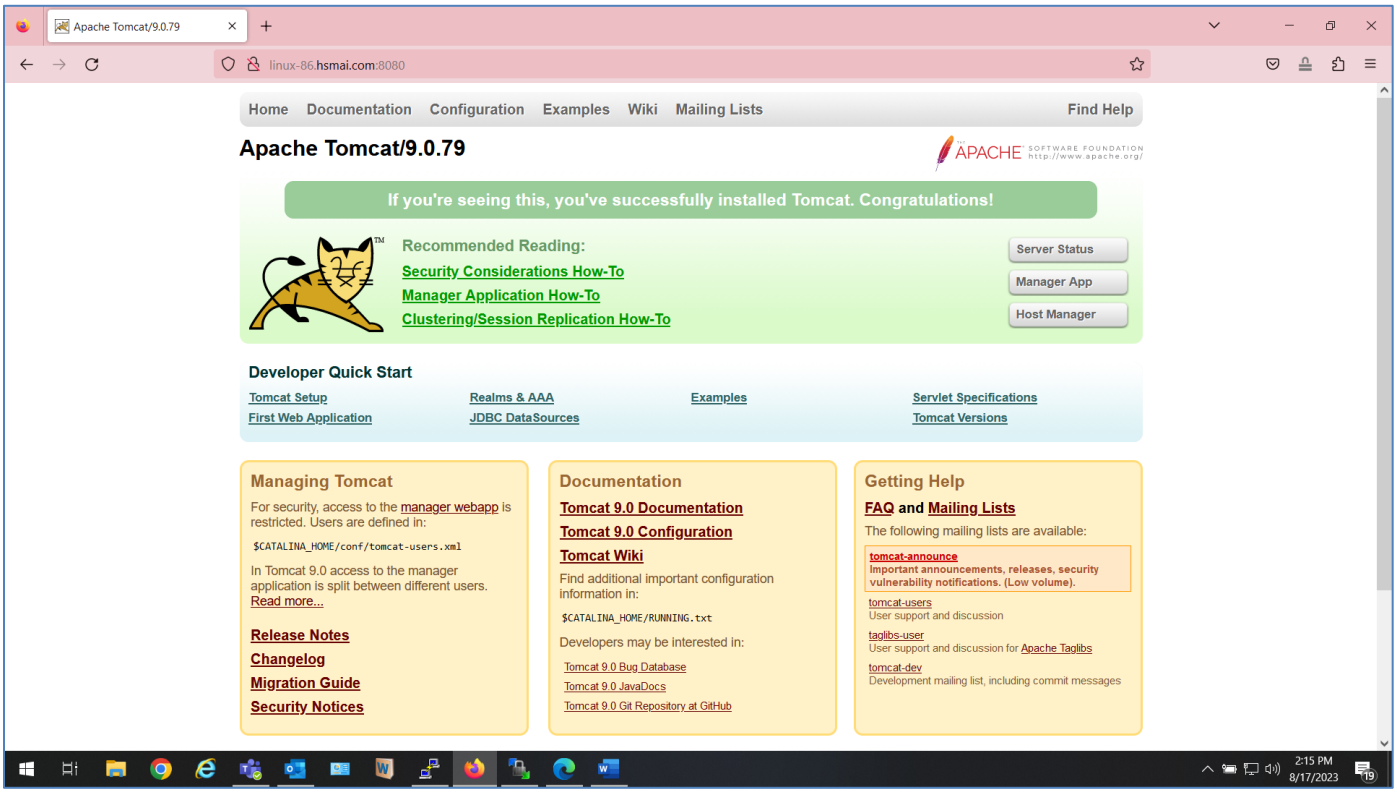
If you're looking for enhanced security through native libraries and the APR connector, consider the Gem Engine toolkit from Thales Customer Support. This toolkit employs OpenSSL engine for advanced cryptographic operations. Familiarize yourself with OpenSSL by consulting the OpenSSL Documentation for insights.

Set up Apache Tomcat

To begin using Apache Tomcat, you need to set it up on your target machines. For detailed instructions, refer to http://tomcat.apache.org/

> **NOTE:** A compatible JDK version must be installed before Apache Tomcat installation. Refer to Apache Tomcat documentation for specific JDK version requirements.

To ensure that Apache Tomcat is up and running post installation, access the following URL: http://<hostname or IP address>:8080/. This URL should display the Apache Tomcat welcome page, confirming that your installation was successful.



Integrating Apache Tomcat with JDK 11 by generating new SSL certificate and key

To integrate Apache Tomcat using JDK 11, you need to follow these steps:

> Configure Java for Luna HSM

> Generate key materials on Luna HSM

> Configure SSL for Apache Tomcat

Configure Java for Luna HSM

Apache Tomcat uses Java JSSE for SSL/TLS support. Configure Java to add support for Luna Provider that will be consumed by Apache Tomcat for securing the SSL keys and certificates on Luna HSM. To configure Luna Provider in Java 11:

1. Log on to Apache Tomcat server as root or as another user having administrative privileges.
2. Ensure that **JAVA_HOME** and **PATH** variables are set. If not, set **JAVA_HOME** and **PATH** variables.

   ```
   # export JAVA_HOME=<JDK_installation_directory>
   # export PATH=$JAVA_HOME/bin:$PATH
   ```

   NOTE: For Windows, set the **JAVA_HOME** and **PATH** System variables under **System**> **Advanced system settings**> **Environment Variables…**

3. Edit the Java Security Configuration file **java.security** located in the directory **<JDK_installation_directory>/conf/security** and add the Luna Provider to the **java.security** file, as demonstrated in the example below:

   ```
   security.provider.1=SUN
   security.provider.2=com.safenetinc.luna.provider.LunaProvider
   security.provider.3=SunRsaSign
   security.provider.4=SunEC
   security.provider.5=SunJSSE
   security.provider.6=SunJCE
   security.provider.7=SunJGSS
   security.provider.8=SunSASL
   security.provider.9=XMLDSig
   security.provider.10=SunPCSC
   security.provider.11=JdkLDAP
   security.provider.12=JdkSASL
   security.provider.13=SunPKCS11
   ```

Generate key materials on Luna HSM

When integrating Java with the Luna Provider, you can establish keys and certificate in the keystore that are linked to a Luna HSM partition. To create keys and certificate in Luna HSM:

1. Create a keystore configuration file named **lunastore**. Include the following entry, where <Partition Name> corresponds to your Luna HSM partition label:

   ```
   tokenlabel:<partition_label>
   ```

   Save the file in the **<Tomcat_Installation>/conf** directory.

2. Generate a key pair in the keystore using the Java **keytool** utility. The key pair will be generated on the registered partition of Luna HSM.

   ```
   # keytool -genkeypair -alias <key label> -keyalg <key algorithm> -keysize <size of key> -sigalg <signing algorithm> -keypass <key password> -keystore <keystore name> -storetype Luna -storepass <partition password> -providerpath
   ```

```
<lunaprovider jar file> -providerclass <luna class path> -J-
Djava.library.path=<luna JSP lib path> -J-cp -J<lunaprovider jar file>
```

Where `storepass` is the Luna HSM partition password.

For example:

```
# keytool -genkeypair -alias lunakey -keyalg RSA -keysize 2048 -sigalg
SHA256withRSA -keypass userpin1 -keystore lunastore -storetype Luna -storepass
userpin1 -providerpath "/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -
providerclass com.safenetinc.luna.provider.LunaProvider -J-
Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
```

When prompted, enter the details to generate key and certificate. Key pair and certificate will be generate on the Luna HSM.

3. View the generated key materials by using the following command and provide the keystore password:

```
# keytool -list -v -keystore lunastore –storetype Luna -providerpath <luna jar
file location> -providerclass <luna class path> -J-Djava.library.path=<luna JSP
lib path> -J-cp -J<lunaprovider jar file>
```

For Example:

```
# keytool -list -v -keystore lunastore -storetype Luna -providerpath
"/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
com.safenetinc.luna.provider.LunaProvider -J-
Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
```

Enter the keystore password, when prompted.

4. Generate a certificate request from a key in the keystore. The system will prompt you for the keystore password.

```
# keytool -certreq -alias lunakey -sigalg SHA256withRSA -file certreq_file -
keystore lunastore -storetype Luna -providerpath
"/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
com.safenetinc.luna.provider.LunaProvider -J-
Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
```

Enter the keystore password, when prompted. File **certreq_file** will be generated in the current directory.

5. Submit the CSR file to your Certification Authority (CA). The CA will authenticate the request and return a signed certificate or a certificate chain. Save the reply and the root certificate of the CA in the current working directory.

6. Import the CA's Root certificate and signed certificate or certificate chain in to the keystore. To import the CA root certificate, execute the following comands:

```
# keytool -trustcacerts -importcert -alias rootca -file root.cer -keystore
lunastore -storetype Luna -providerpath
"/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
com.safenetinc.luna.provider.LunaProvider -J-
Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
```

To import the signed certificate reply or certificate chain, execute the following command:

```
# keytool -trustcacerts -importcert -alias lunakey -file certchain.p7b -
keystore lunastore -storetype Luna -providerpath
"/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
```

```
com.safenetinc.luna.provider.LunaProvider -J-
Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
```

Enter the keystore password, when prompted. Here, **root.cer** and **certchain.p7b** are the CA Root Certificate and Signed Certificate Chain, respectively.

7. Edit the Java Security Configuration file **java.security** located in the directory **<JDK_installation_directory>/conf/security** and add the Luna Provider to the **java.security** file, as demonstrated in the example below:

```
security.provider.1=SUN
security.provider.2=SunEC
security.provider.3=SunJSSE
security.provider.4=SunJCE
security.provider.5=SunJGSS
security.provider.6=SunSASL
security.provider.7=XMLDSig
security.provider.8=SunPCSC
security.provider.9=JdkLDAP
security.provider.10=JdkSASL
security.provider.11=SunPKCS11
security.provider.12=com.safenetinc.luna.provider.LunaProvider
security.provider.13=SunRsaSign
```

Configure SSL for Apache Tomcat

SSL configuration for Apache Tomcat involves setting up the SSL key and certificate stored in the keystore for secure communication. Follow these steps to configure SSL in the server.xml file located in the **<Tomcat_installation_directory>/conf** directory:

1. Stop the server, if it is running. Execute the **shutdown.bat** (for Windows) or **shutdown.sh** (for Unix) script found in the **bin** folder of your Tomcat installation directory.
2. Open the **server.xml** file of Tomcat server. You can either uncomment the existing connector and update it, or you can add the following code snippet as a new connector configuration without modifying the existing one:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
  sslImplementationName="org.apache.tomcat.util.net.jsse.JSSEImplementation"
           maxThreads="150" scheme="https" secure="true" SSLEnabled="true"
           clientAuth="false" sslProtocol="TLS"
           keystoreType="Luna" keystoreFile="conf/lunastore" keyAlias="lunakey"
           keystorePass="userpin1" />
```

3. Save the **server.xml** file and close the text editor. Ensure that the keystore settings match your environment specifications accurately.

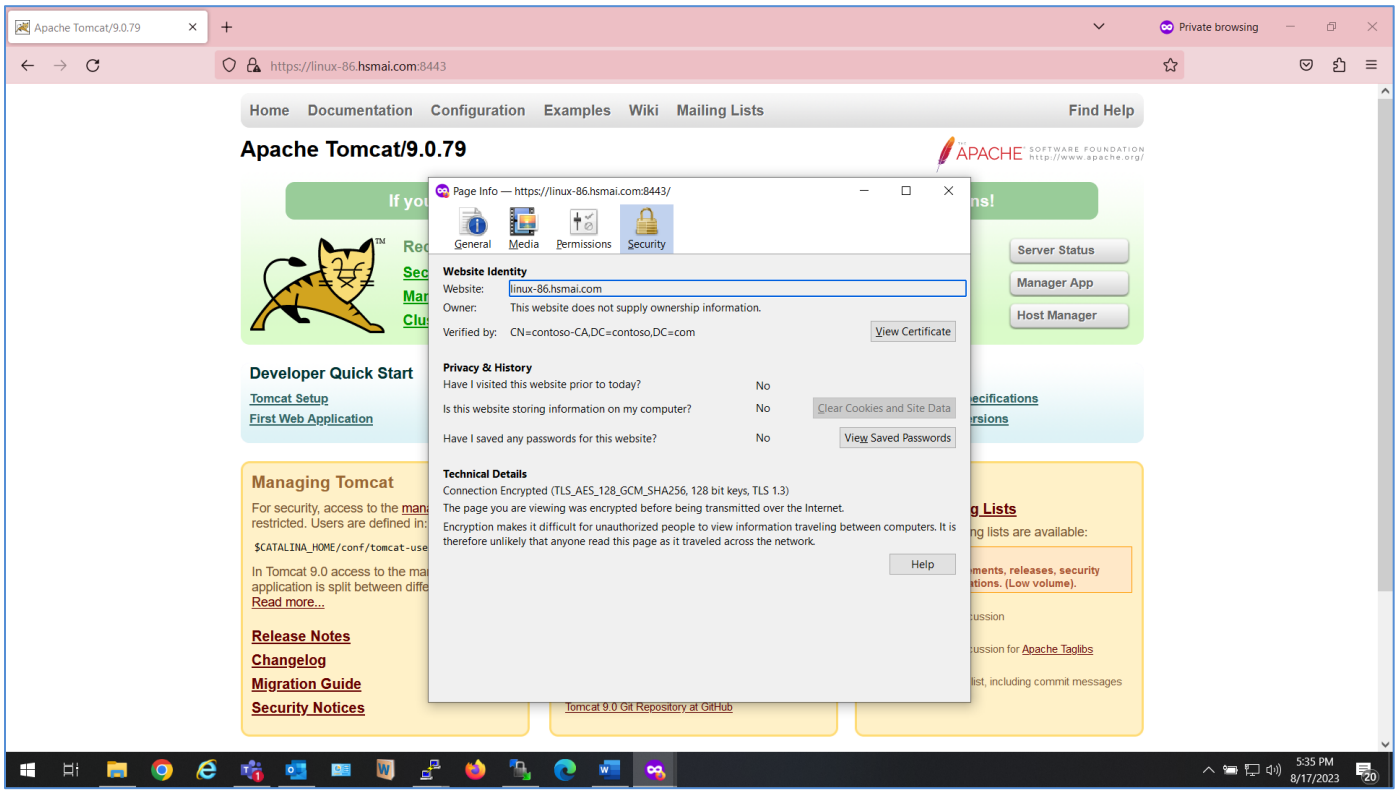
4. Create a file **setenv.sh** in **$CATALINA_HOME/bin** folder with the following entries:

```
#!/bin/sh

export CLASSPATH=/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar

export CATALINA_OPTS=-Djava.library.path=/usr/safenet/lunaclient/jsp/lib/
```

5. Start the Tomcat server using the batch file **startup.bat** or **startup.sh** under the **bin** directory of **<Tomcat_installation_directory>**.
6. If the Tomcat starts successfully, you should be able to see the default page of Tomcat on the browser using https and port 8443:

   https://<hostname or IP Address>:8443/

   The SSL certificate will be the same as the one that you generated and stored in Luna Keystore.



This completes the Apache Tomcat integration with Luna HSM and SSL certificate private key is secured on HSM partition. The SSL page will be accessible only if HSM partition is accessible and available to Apache Tomcat Server

Integrating Apache Tomcat with JDK 8 by generating or migrating SSL certificate and key

To integrate Apache Tomcat with JDK 8, you need to follow either of these use cases, depending on your requirement:

> Integrate Apache Tomcat by generating new SSL certificate and key on Luna HSM

> Integrate Apache Tomcat by migrating existing SSL Certificate and key to Luna HSM

Integrate Apache Tomcat by generating new SSL certificate and key on Luna HSM

For integrating JDK 8 Compatible Apache Tomcat through generation of New SSL Certificate and Key, you need to:

> Configure Java for Luna HSM

> Generate key materials on Luna HSM

> Configure SSL for Apache Tomcat

Configure Java for Luna HSM

Apache Tomcat uses Java JSSE for SSL/TLS support. Configure Java to add support for Luna Provider that will be consumed by Apache Tomcat for securing the SSL keys and certificates on Luna HSM. To configure Luna Provider in Java 8:

1. Log on to Apache Tomcat server as root or as another user having administrative privileges. Ensure that **JAVA_HOME** and **PATH** variables are set. If not, set **JAVA_HOME** and **PATH** variables.

   ```
   # export JAVA_HOME=<JDK_installation_directory>
   # export PATH=$JAVA_HOME/bin:$PATH
   ```

   NOTE: For Windows, set the **JAVA_HOME** and **PATH** System variables under **System**> **Advanced system settings**> **Environment Variables…**

2. Edit the Java Security Configuration file **java.security** located in the directory **<JDK_installation_directory>/jre/lib/security** and add the Luna Provider to the **java.security** file as demonstrated in the example below:

   Example:

   ```
   security.provider.1=sun.security.provider.Sun
   security.provider.2=sun.security.rsa.SunRsaSign
   security.provider.3=sun.security.ec.SunEC
   security.provider.4=com.sun.net.ssl.internal.ssl.Provider
   security.provider.5=com.sun.crypto.provider.SunJCE
   security.provider.6=com.safenetinc.luna.provider.LunaProvider
   security.provider.7=sun.security.jgss.SunProvider
   security.provider.8=com.sun.security.sasl.Provider
   security.provider.9=org.jcp.xml.dsig.internal.dom.XMLDSigRI
   security.provider.10=sun.security.smartcardio.SunPCSC
   security.provider.11=sun.security.mscapi.SunMSCAPI
   ```

3. Copy the **LunaAPI.dll** (Windows) or **libLunaAPI.so** (UNIX) and **LunaProvider.jar** file from the **<Luna_installation_directory>/jsp/lib** folder to the **<JDK_installation_directory>/jre/lib/ext** directory.

Generate key materials on Luna HSM

When Java is configured to use Luna Provider, we can create the keys and certificate in the keystore pointing to Luna HSM partition. To create keys and certificate in Luna HSM:

1. Create a keystore config file named **lunastore** and add the following entry where <Partition Name> would be your Luna HSM partition label:

   ```
   tokenlabel:<partition_label>
   ```

   Save the file, preferably in the **<Tomcat_Installation>/conf** directory.

2. Generate a key pair in the keystore using the Java **keytool** utility. The key pair will be generated on the registered partition of Luna HSM.

   ```
   # keytool -genkeypair -alias <key label> -keyalg <key algorithm> -keysize <size of key> -sigalg <signing algorithm> -keypass <partition password> -keystore <keystore name> -storepass <partition password> -storetype <luna keystore>
   ```

   For example:

   ```
   # keytool -genkeypair -alias lunakey -keyalg RSA -keysize 2048 -sigalg SHA256withRSA -keypass userpin1 -keystore lunastore -storepass userpin1 -storetype luna
   ```

   Enter the details to generate key and certificate in the Luna HSM and keystore in the current directory.

3. To display the generated key materials, use the following command:

   ```
   # keytool -list -v -storetype luna -keystore lunastore
   ```

4. Generate a certificate request from a key in the keystore. The system will prompt you for the keystore password.

   ```
   # keytool -certreq -alias lunakey -sigalg SHA256withRSA -file certreq_file -storetype luna -keystore lunastore
   ```

   Enter the keystore password, when prompted. File **certreq_file** will be generated in the current directory.

5. Submit the CSR file to your Certification Authority (CA). The CA will authenticate the request and return a signed certificate or a certificate chain. Save the reply and the root certificate of the CA in the current working directory.

6. Import the CA's Root certificate and signed certificate or certificate chain in to the keystore. To import the CA root certificate, execute the following:

   ```
   # keytool -trustcacerts -importcert -alias rootca -file root.cer -keystore lunastore -storetype luna
   ```

   To import the signed certificate reply or certificate chain, execute the following:

   ```
   # keytool -trustcacerts -importcert -alias lunakey -file certchain.p7b -keystore lunastore -storetype luna
   ```

   Here, **root.cer** and **certchain.p7b** are the CA Root Certificate and Signed Certificate Chain, respectively.

Configure SSL for the Apache Tomcat

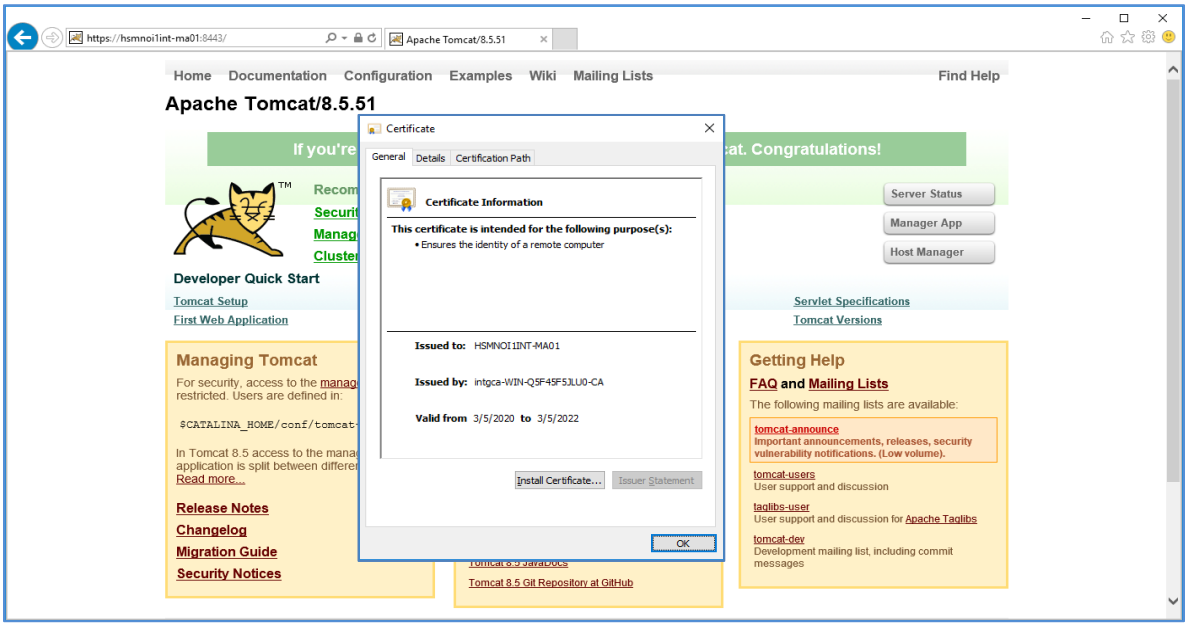
Apache Tomcat server uses the SSL key and certificate stored in the keystore for SSL communication. Apache Tomcat uses **server.xml** file available in **<Tomcat_installation_directory>/conf** to define connector setting for SSL. To configure SSL for Apache Tomcat:

1. Stop the Tomcat server if it is running. Execute the **shutdown.bat** (for Windows) or **shutdown.sh** (for Unix) script found in the **bin** folder of your Tomcat installation directory.

2. Modify the **server.xml** file of the Tomcat server. You can uncomment and update the existing connector configuration, or you can add the following snippet as a new connector configuration without modifying the existing one.

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
    sslImplementationName="org.apache.tomcat.util.net.jsse.JSSEImplementation"
    maxThreads="150" scheme="https" secure="true" SSLEnabled="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreType="Luna" keystoreFile="conf/lunastore" keyAlias="lunakey"
    keystorePass="userpin1" />
```

3. Save the **server.xml** file and close the text editor. Ensure that the keystore settings match your environment specifications accurately.
4. Start the Tomcat server by running the appropriate batch file, either **startup.bat** for Windows or **startup.sh** for Unix/Linux. You can find these files in the **bin** directory of your Tomcat installation directory.
5. Once the Tomcat server is running successfully, open a web browser and enter the following URL: https://<hostname or IP Address>:8443/.
    a. Replace <hostname or IP Address> with the actual hostname or IP address of your server.
    b. This URL will allow you to access the default page of Tomcat using HTTPS (secure) protocol on port 8443.
    c. The SSL certificate used for the connection will be the same one that you generated and stored in Luna Keystore.



This completes the Apache Tomcat integration with Luna HSM and SSL certificate private key is secured on HSM partition. The SSL page will be accessible only if HSM partition is accessible and available to Apache Tomcat Server.

Integrating Apache Tomcat by migrating existing SSL certificate and key to Luna HSM

For integrating JDK 8 compatible Apache Tomcat through migration of an existing SSL certificate and key, you need to:

> Configure Java for Luna HSM

> Migrate key materials from JKS to Luna Keystore

> Reconfigure SSL for the Apache Tomcat

NOTE: Before proceeding, it is assumed that you have installed Apache Tomcat and have configured the SSL using the key and certificate available on Java Keystore.

Configure Java for Luna HSM

To configure Java for Apache Tomcat for securing the SSL keys and certificates on Luna HSM, refer to the Configure Java for Luna HSM section.

Migrate key materials from JKS to Luna Keystore

After configuring Java to use Luna Provider, you can migrate the keys and certificate from JKS to Luna Keystore by following these steps:

1. Create a keystore config file named **lunastore** and add the following entry where <Partition Name> would be your Luna HSM partition label:

   ```
   tokenlabel:<partition_label>
   ```

   Save the file, preferably in the **<Tomcat_Installation>/conf** directory.

2. Migrate the Java keystore to Luna keystore including SSL certificate/key using the **keytool** utility. The certificate/key will be migrated on the registered partition of Luna HSM.

   ```
   # keytool -importkeystore -srckeystore <source keystore name> -srcstorepass <source keystore password> -srcalias <source key label> -destalias <destination key label> -destkeystore <destination keystore name> -deststorepass <partition password> -deststoretype <luna keystore>
   ```

   For Example:

   ```
   # keytool -importkeystore -srckeystore mykeystore.jks -srcstorepass changeit -srcalias tomcat_key -destalias tomcat_migrated_key -destkeystore lunastore -deststorepass userpin1 -deststoretype luna
   ```

   Provide partition password, when prompted.

3. View the generated key materials by using the following command:

   ```
   # keytool -list -v -alias tomcat_migrated_key -storetype luna -keystore lunastore
   ```

   Provide partition password, when prompted.

NOTE: It is recommended that you should destroy the Java keystore after migrating the key materials to Luna keystore. Keeping the SSL key in software keystore may result in security breach.

Reconfigure SSL for Apache Tomcat

After successfully migrating the JKS keystore to **lunastore**, you need to reconfigure SSL settings in **server.xml** to pick the SSL certificate/key from **lunastore**. Use the following steps for reconfiguration:

1. Stop the server if it is running. Run the **shutdown.bat** or **shutdown.sh** script provided under **bin** folder of **<Tomcat_installation_directory>**.
2. Edit the **server.xml** file of Tomcat server as follows:

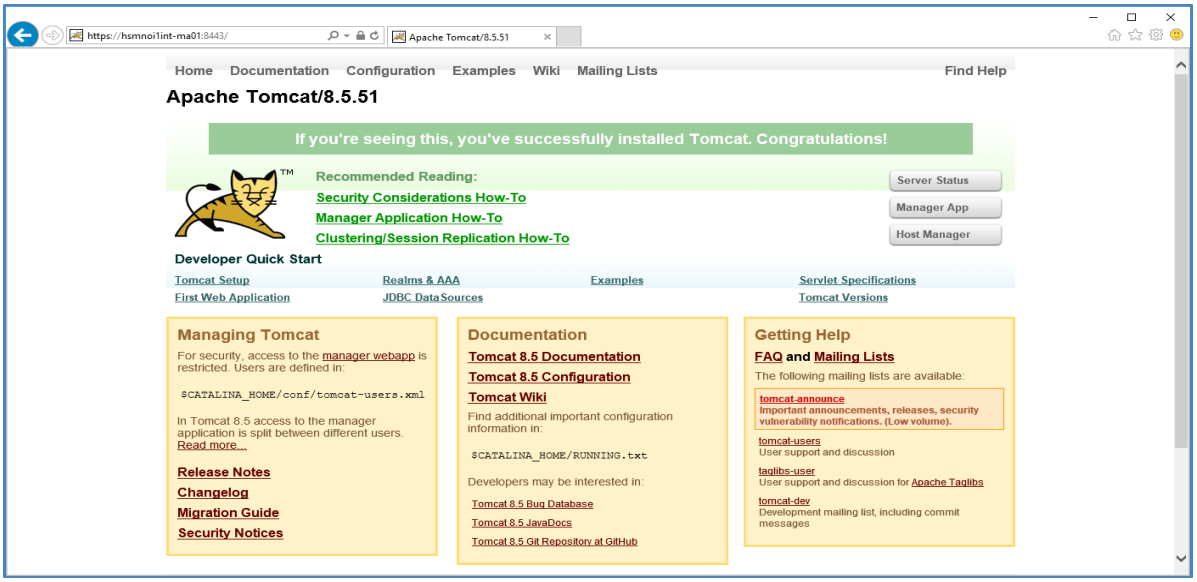
```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
    sslImplementationName="org.apache.tomcat.util.net.jsse.JSSEImplementation"
    maxThreads="150" scheme="https" secure="true" SSLEnabled="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreType="Luna" keystoreFile="conf/lunastore"
    keyAlias="tomcat_migrated_key"  keystorePass="userpin1" />
```

   Ensure that the keystore values are correct as per your environment.

3. Start the Tomcat server using the batch file **startup.bat** or **startup.sh** under the **bin** directory of **<Tomcat_installation_directory>**. If Tomcat starts successfully, you should be able to see the default page of Tomcat on the browser using https and port 8443:

   https://<hostname or IP Address>:8443/

   The SSL certificate will be identical to the one that you migrated and stored in Luna Keystore.



Integrating Luna HSM with Apache Tomcat using Native Library

This document contains detailed instructions and procedures to integrate Luna HSM with Apache Tomcat using Native Library and APR connector. This integration contains the following topics:

Install Gem Engine and OpenSSL

To install Gem Engine and OpenSSL:

1. Install the **make** and **gcc** compiler packages.
2. Create a directory to store all the files you'll download for the installation.

   ```
   # mkdir -p /export/home
   ```

3. Download OpenSSL and save it in the directory you just created. You can use **wget** for this.

   ```
   # wget https://www.openssl.org/source/openssl-1.1.1k.tar.gz
   ```

   NOTE: If you want to use the HSM in Non-FIPS mode, OpenSSL version 1.1.1x is suitable. For FIPS mode HSM, go for OpenSSL version 1.0.2x with the FIPS module.

4. Extract the Gem Engine toolkit in the /export/home directory. For Example:

   ```
   # tar xf 610-012987-004_SW_OPENSSL_TOOLKIT_GemEngine_v1.4_RevA.tar
   ```

5. Rename the gemengine-1.4 directory to gemengine.

   ```
   # mv gemengine-1.4/ gemengine
   ```

6. Extract the OpenSSL source files from the downloaded tarball. For Example:

   ```
   # tar xvfz openssl-1.1.1k.tar.gz
   ```

7. Rename the directory openssl-1.1.1k to openssl.

   ```
   # mv openssl-1.1.1k openssl
   ```

8. Go to the gemengine directory you previously created.

   ```
   # cd /export/home/gemengine/
   ```

9. Run the gembuild config command with the specified options:

   ```
   # ./gembuild config --openssl-source=/export/home/openssl --prefix=/usr/local --config-bits=64
   ```

   NOTE: If using OpenSSL version 1.0.2x, add the `--compat-102` option to the above command.

10. Build the OpenSSL libraries using the following command.

```
# ./gembuild openssl-build
```

> **NOTE:** For OpenSSL version 1.0.2x, build and install OpenSSL FIPS module before running this step. Refer to the gemengine/docs/README-GEMBUILD file for detailed steps.

11. Install the OpenSSL libraries using the following command:

```
# ./gembuild openssl-install
```

12. Export the PATH and LD_LIBRARY_PATH environment variables:

```
# export PATH=/usr/local/ssl/bin:$PATH
# export LD_LIBRARY_PATH=/usr/local/ssl/lib:$LD_LIBRARY_PATH
```

13. Verify the installed OpenSSL version.

```
# openssl version -a
```

14. Compile the Gem dynamic engine.

```
# ./gembuild engine-build
```

15. Install the Gem dynamic engine.

```
# ./gembuild engine-install
```

16. Run the following command to verify that the GemEngine is correctly installed.

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

The output will be:

```
(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
```

17. Open the /etc/Chrystoki.conf file and add the following GemEngine section. Replace `slot_label` with the appropriate partition label.

```
GemEngine = {
   LibPath64 = /opt/safenet/lunaclient/lib/libCryptoki2_64.so;
   LibPath = /opt/safenet/lunaclient/lib/libCryptoki2_64.so;
   EnableDsaGenKeyPair = 1;
   EnableRsaGenKeyPair = 1;
   DisablePublicCrypto = 1;
   EnableRsaSignVerify = 1;
   EnableLoadPubKey = 1;
   EnableLoadPrivKey = 1;
   DisableCheckFinalize = 1;
   IntermediateProcesses = 0;
   DisableEcdsa = 1;
   DisableDsa = 0;
   DisableRand = 0;
   DisableSessionCache = 0;
   EngineInit = "<slot_label>":0:0:passfile=/tmp/passfile;
   EnableLoginInit = 1;
   }
```

18. Create a text file and store the partition password in it.

```
# echo <partition_password> > /tmp/passfile
```

19. Test the gem engine by running:

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

The output should indicate that the Gem engine is available.

Install Apache Tomcat

To install Apache Tomcat:

1. Download and install the supported JAVA version for Tomcat.
2. Export JAVA_HOME environment variable.

```
# export JAVA_HOME=/export/home/jdk1.8.0_152/
```

3. Create a tomcat group:

```
# groupadd tomcat
```

4. Create a tomcat user and set its password.

```
# useradd -s /bin/bash -g tomcat -m tomcat
# passwd tomcat
```

5. Create a directory where Tomcat will be installed.

```
# mkdir /usr/local/tomcat9
```

6. Go the /usr/local/tomcat9/ directory.

```
# cd /usr/local/tomcat9/
```

7. Download the Apache Tomcat tarball.

```
# wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.20/bin/apache-tomcat-9.0.20.tar.gz
```

8. Extract the downloaded tarball.

```
# tar -xf apache-tomcat-9.0.20.tar.gz
```

9. Change the ownership of /usr/local/tomcat9/ directory to tomcat user and group.

```
# chown -R tomcat:tomcat /usr/local/tomcat9/
```

10. Export CATALINA_HOME and CATALINA_BASE environment variable.

```
# export CATALINA_HOME="/usr/local/tomcat9
# export CATALINA_BASE="/usr/local/tomcat9"
```

Install Apr and Apr-util

To install apr and apr-util:

1. Go to the /export/home/ directory.

```
# cd /export/home/
```

2. Download the apr tarball.

```
# wget https://apachemirror.wuchna.com/apr/apr-1.7.0.tar.bz2
```

3. Extract the downloaded tarball.

```
# tar -zxvf  apr-1.7.0.tar.gz
```

4. Go to the apr-1.7.0 directory

```
# cd apr-1.7.0/
```

5. Create a blank file.

```
# touch libtoolT
```

6. Run the configure command.

On RHEL 7/8

```
# ./configure
```

On Solaris 11

```
# CFLAGS="-m64" ./configure
```

7. Run make and make install.

```
# make
# make install
```

8. Go to the /export/home/ directory.

```
# cd /export/home/
```

9. Download the apr-util tarball.

```
# wget https://apachemirror.wuchna.com/apr/apr-util-1.6.1.tar.gz
```

10. Extract the downloaded tarball.

```
# tar -zxvf  apr-util-1.6.1.tar.gz
```

11. Run the configure command.

On RHEL 7/8

```
# ./configure --with-apr=/usr/local/apr
```

On Solaris 11

```
# CFLAGS="-m64" ./configure --with-apr=/usr/local/apr
```

12. Run the make and make install commands.

```
# make
# make install
```

13. Export LD_LIBRARY_PATH environment variable.

```
# export LD_LIBRARY_PATH=/usr/local/apr/lib:$LD_LIBRARY_PATH
```

Install Tomcat Native

To install Tomcat native:

1. Go to /usr/local/tomcat9/bin.

```
# cd /usr/local/tomcat9/bin
```

2. Download tomcat-native-1.2.28-src tarball if it is not present.

3. Extract the downloaded tarball.

```
# tar -zxvf tomcat-native.tar.gz
```

4. Go to tomcat-native-1.2.28-src/native/ directory.

```
# cd tomcat-native-1.2.28-src/native/
```

5. Run the configure command.

On RHEL 7/8

```
# ./configure --with-apr=/usr/local/apr --with-java-
home=/export/home/jdk1.8.0_152/ --with-ssl=/usr/local/ssl/
```

On Solaris 11

```
# CFLAGS="-m64" ./configure --with-apr=/usr/local/apr --with-java-
home=/export/home/jdk1.8.0_152/ --with-ssl=/usr/local/ssl/
```

6. Run make and make install commands.

```
# make
```

```
# make install
```

7. Create a file /usr/local/tomcat9/bin/setenv.sh and add the following line.

```
export LD_LIBRARY_PATH=/usr/local/ssl/lib:/usr/local/apr/lib:
/usr/local/tomcat9/lib:$LD_LIBRARY_PATH
```

Configure SSL in Apache Tomcat

Follow these steps to configure SSL in Apache Tomcat:

1. Execute the following command to generate keys using Luna HSM. This will also save the certificate request and key reference.

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

2. Run the following command to verify the generated key pair on Luna HSM. You'll be prompted to enter the partition password.

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

3. Submit the CSR file to a CA. Once authenticated, the CA will provide a signed certificate or certificate chain. Save the CA-signed certificate with name servercert.cer in the system directory.

4. For the purpose of demonstration, create a self-signed certificate servercert.cer using a test key server.key:

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

```
# openssl req -engine gem -new -x509 -days 365 -key server.key -out
servercert.cer
```

> **NOTE:** This integration uses self-signed certificates in a test environment only. For a production environment, we recommend using a more secure method such as a certificate authority to issue the certificate.

5. Copy the generated **server.key** and **servercert.cer** files to the Tomcat configuration directory.

```
# cp server.key /usr/local/tomcat9/conf/
```

```
# cp servercert.cer /usr/local/tomcat9/conf/
```

6. Open /usr/local/tomcat9/conf/server.xml file and make the following changes.

   a. Add the SSLEngine="gem" to the following listener.

```
<Listener className="org.apache.catalina.core.AprLifecycleListener"
SSLEngine="gem" />
```

   b. Add the following to enable SSL support.

```
  <Connector
        protocol="org.apache.coyote.http11.Http11AprProtocol"
        port="8443"
        maxThreads="150"
        SSLEnabled="true" >
      <SSLHostConfig>
        <Certificate
            certificateKeyFile="/usr/local/tomcat9/conf/server.key"
            certificateFile="/usr/local/tomcat9/conf/servercert.cer"
            type="RSA"
            />
      </SSLHostConfig>
    </Connector>
```

7. Change the ownership of /usr/local/tomcat9/ directory to tomcat user and group.

```
# chown -R tomcat:tomcat /usr/local/tomcat9/
```

8. Start the Tomcat service.

```
# /usr/local/tomcat9/bin/catalina.sh start
```

9. Open any web browser and access the page over SSL

```
https://<apache_tocat_server_ip>:8443
```

Run Apache Tomcat with non-root user

To run Apache Tomcat with non-root user:

1. Stop the Apache Tomcat server if it is running.

```
# /usr/local/tomcat9/bin/catalina.sh stop
```

2. Log in as non-root user, that is, tomcat user.
3. Export the following environment variables.

```
# export JAVA_HOME=/export/home/jdk1.8.0_152

# export CATALINA_HOME="/usr/local/tomcat9

# export CATALINA_BASE="/usr/local/tomcat9"

# export PATH=/export/home/jdk1.8.0_152/bin:/usr/local/ssl/bin:$PATH

# export LD_LIBRARY_PATH=/usr/local/ssl/lib:/usr/local/apr/lib:
/usr/local/tomcat9/lib:$LD_LIBRARY_PATH
```

4. Start the Tomcat service.

```
# /usr/local/tomcat9/bin/catalina.sh start
```

5. Open any web browser and access the page over SSL

```
https://<apache_tocat_server_ip>:8443
```

This completes the integration of Luna HSM with Apache Tomcat using native library and apr connector.

Troubleshooting

Problem

The following error may be encountered after running `/gembuild openssl-build` on Solaris 11:

```
sh: line 1: cc: not found
*** Error code 127
```

Solution

To resolve this issue:

a. Go to gemengine directory and open gembuild file.

b. Go to line no. 339 and make the following changes:

```
#     LUNA_CFG_TGT=solaris64-sparcv9-cc
      LUNA_CFG_TGT=solaris64-sparcv9-gcc
```

Problem

If you get the following error after running `./gembuild openssl-build` on Solaris 11:

```
unrecognized option '-KPIC'
```

Solution

To resolve this issue:

a. Open gemengine\engine\configure file.

b. Go to line no. 51 and make the following changes:

```
                CFLAGS1="-xarch=v9 -fpic"
                LDFLAGS1="-xarch=v9 -G -h $LDSO -fpic"
```

Contacting Customer Support

If you encounter a problem while installing, registering, or operating this product, 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.