OpenSSL

v3.5

Integration Guide

u.Trust GP HSM Se-Series

utimaco®

Imprint

Copyright 2025	Utimaco IS GmbH Germanusstr. 4 D-52080 Aachen Germany
Phone	AMERICAS +1-844-UTIMACO (+1 844-884-6226) EMEA +49 800-627-3081 APAC +81 800-919-1301
Internet	https://support.hsm.utimaco.com/
e-mail	support@utimaco.com
Document Version	1.0.0
Date	2025-08-29
Status	PUBLISHED
Document No.	IG-2025-0028

Table of Contents

1 Introduction

This guide is part of the information and support provided by Utimaco. Additional documentation produced to support your Utimaco u.trust Anchor or Quantum Protect products can be found in the document directory of the Utimaco u.trust Anchor product bundle or Quantum Protect download, respectively. All Utimaco u.trust Anchor product documentation is available from Utimaco's website at https://utimaco.com/.

1.1 About This Guide

This guide provides an integration explanation, explaining how to integrate a Utimaco u.trust Anchor Hardware Security Module (HSM) with a Quantum Protect module and OpenSSL.

1.2 Target Audience

This guide is intended for administrators of OpenSSL and of Utimaco HSMs.

1.3 Purpose of the Integration

The integration of OpenSSL with Quantum Protect is designed to bridge the gap between the growing demand for post-quantum security and the practical needs of Linux-based applications that rely on OpenSSL as their cryptographic backbone.

With version 3.5, OpenSSL introduced support for the Post-Quantum algorithms standardized by NIST. This milestone is significant because OpenSSL is the most widely used cryptographic library in Linux environments, serving as the foundation for countless applications and services. Incorporating post-quantum cryptography (PQC) into OpenSSL allows these applications to adopt next-generation algorithms without requiring extensive redesign or migration efforts.

However, OpenSSL does not natively integrate with Hardware Security Modules (HSMs) or other secure key storage mechanisms. Instead, it relies on external providers, such as the PKCS#11 provider, to enable secure hardware-backed key management. This limitation makes it necessary to combine OpenSSL's cryptographic capabilities with solutions like Quantum Protect, which extend the u.trust HSM to support PQC.

This integration combines the strengths of OpenSSL's new quantum-ready capabilities with the security of u.trust HSM. It enables Linux applications to use post-quantum cryptographic keys seamlessly and securely, ensuring long-term resilience against emerging quantum threats.

1.4 Document Conventions

The following conventions are used in this guide:

Convention	Use	Example
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press the **OK** button.
`Monospaced`	File names, folder and directory names, commands, file outputs, programming code samples	You will find the file `example.conf` in the `/exmp/demo/` directory.
Italic	References and important terms	See Chapter 3, "Sample Chapter", in the *u.trust Anchor - csadm Manual* or [CSADM].

Table 1: Document Conventions

Special icons are used to highlight the most important notes and information.

Here you find important safety information that should be followed.

Here you find additional notes or supplementary information.

This message marks the result expected after the successful execution of an instruction.

1.5 Abbreviations

The following abbreviations are used in this guide:

Abbreviation	Meaning
CA	Certificate Authority
CAT	CryptoServer Administration Tool
CD	Compact Disc
CSR	Certificate Signing Request
GUI	Graphical User Interface
HSM	Hardware Security Module
IP	Internet Protocol
LAN	Local Area Network
PCIe	PCI Express Interface
PKCS#11	Public-Key Cryptography Standard #11
RSA	Rivest-Shamir-Adleman
SO	Security Officer
SSL	Secure Socket Layer
TLS	Transport Layer Security

Abbreviation	Meaning
URL	Uniform Resource Locator

Table 2: List of Abbreviations

2 Overview

2.1 OpenSSL

OpenSSL is an open-source tool for using the Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols for Web authentication. It offers cryptographic functions to support SSL/TLS protocols.

It allows users to perform various SSL-related tasks, including generating CSR (Certificate Signing Request) and private keys, and installing SSL certificates. Most Linux distributions come with pre-compiled OpenSSL, but if you are on a Windows system, you can install it manually.

OpenSSL has an abstraction layer called providers, which can delegate cryptographic operations to different pieces of software or hardware.

The pkcs11-provider provides an OpenSSL plugin that tries to fit the PKCS#11 API within the OpenSSL provider API. That is, it provides a gateway between PKCS#11 modules and the OpenSSL provider API.

2.2 Utimaco u.trust Anchor GP HSM

The u.trust Anchor is a next-generation hardware security module developed by Utimaco IS GmbH. It is a multi-tenant, physically protected, and tamper-resistant cryptographic appliance designed to perform high-assurance cryptographic operations and manage cryptographic keys securely. The u.trust General Purpose HSM is built on a modern, container-based design inspired by cloud technology. With support for up to 31 containers and multiple PKCS #11 partitions per cHSM, it ensures seamless application separation and key partitioning, making it an ideal choice for all types of cryptographic applications. It's also upgradeable for specific use cases like blockchain and 5G, and offers flexibility for custom solutions, including proprietary algorithms and customer key derivations via the Software Development Kit.

2.3 Utimaco Quantum Protect

Quantum Protect (QP) is an extension to the Utimaco u.trust SecurityServer General Purpose Hardware Security Modules (HSMs).

It extends the cryptographic algorithms running inside the HSMs' tamper-proof environment, adding support for different “Post-Quantum” algorithms.

3 Integration Requirements and Prerequisites

Ensure the system environment you will be using meets the following hardware and software requirements.

This guide assumes that the user has already installed and configured the required Software.

3.1 Tested Versions

These integrations have been successfully tested with the Utimaco HSM and OpenSSL.

Operating System	OpenSSL	Utimaco u.trust Anchor cHSM Version	Utimaco HSM
RHEL 9	OpenSSL 3.5.0	u.trust Anchor cHSM 6.2.0	u.trust Anchor Se-Series
RHEL 10	OpenSSL 3.5.0	u.trust Anchor cHSM 6.2.0	u.trust Anchor Se-Series
Ubuntu 24.04	OpenSSL 3.5.0	u.trust Anchor cHSM 6.2.0	u.trust Anchor Se-Series
Ubuntu 22.04	OpenSSL 3.5.0	u.trust Anchor cHSM 6.2.0	u.trust Anchor Se-Series

Table 3: List of Tested versions

3.2 Hardware and Software Requirements

Hardware	Hardware Requirements
Utimaco u.trust Anchor LAN HSM	u.trust Anchor Se-Series cHSM available with firmware SecurityServer 6.2.0

Table 4: List of Hardware Requirements

Software	Software Requirements
OpenSSL	OpenSSL 3.5.0
pkcs11-provider	1.1.0 from Utimaco
HSM Interface	SecurityServer PKCS#11 Provider
Utimaco Quantum Protect	v1.2.0

Table 5: List of Software Requirements

3.3 Prerequisites

Before you begin, please ensure that you have:

- A u.trust Anchor cHSM setup and configured with SecurityServer-SDK Template. Refer to the u.trust Anchor Se-Series documentation to set up the cHSM.
- An MBK created and stored on each cHSM. Refer to the u.trust Anchor Se-Series documentation to set up the MBK.
- The u.trust Anchor cHSM Default Admin replaced with a new admin user for production environments.
- An operating system listed in Tested Versions.
- A SecurityServer version listed in Tested Versions.
- Familiarized yourself with the OpenSSL documents and setup process.

4 Installation and Configuration

4.1 Setting Up u.trust Anchor HSM

If you have not done so, create and request an Utimaco Support Portal Account. This will allow you to download the software components needed for this installation.

If you have purchased an HSM from Utimaco, you will need the associated product bundle containing the required Windows or Linux software packages. This bundle can be downloaded from the Utimaco Support Portal. Ensure you request access to the product bundle and the Quantum Protect module beforehand through the support portal.

4.1.1 Installing Quantum Protect

To install the Quantum Protect module on the cHSM, do the following:

1. Download the Quantum Protect Software from the Utimaco Support Portal.
2. Ensure you have a running cHSM with the SecurityServer-SDK Template

```
gladm -d <IP> -p <PORT> -u admin -k :cs2:auto:USB0 chsm-list-slots
  1: c0747ffb-592f-4f78-bae6-cad316343129 SecurityServer [regular] - running
  2: 1c6837b7-20b4-430f-83c6-ea97226ae853 SecurityServer-SDK [regular] - running
  3:
  4:
```

3. Ensure there are no VDM-using modules (BRICKS, OSCCA, or other third-party modules).

```
csadm Dev=PORT@IP logonSign=ADMIN,:cs2:auto:USB0 ListFirmware

  ID name         type version        initialization level
-------------------------------------------------------------
   0 SMOS         A32  6.2.0.0        INIT_OK
   4 POST         A32  6.2.0.0        INIT_OK
   a HCE          A32  6.2.0.0        INIT_OK
  68 CXI          A32  6.2.0.0        INIT_OK
  81 VDES         A32  6.2.0.0        INIT_OK
  82 PP           A32  6.2.0.0        INIT_OK
  83 CMDS         A32  6.2.0.0        INIT_OK
  84 VRSA         A32  6.2.0.0        INIT_OK
  85 SC           A32  6.2.0.0        INIT_OK
  86 UTIL         A32  6.2.0.0        INIT_OK
  87 ADM          A32  6.2.0.0        INIT_OK
```

```
 88 DB           A32  6.2.0.0        INIT_OK
 89 HASH         A32  6.2.0.0        INIT_OK
 8a STUN         A32  6.2.0.0        INIT_OK
 8b AES          A32  6.2.0.0        INIT_OK
 8d DSA          A32  6.2.0.0        INIT_OK
 8e LNA          A32  6.2.0.0        INIT_OK
 8f ECA          A32  6.2.0.0        INIT_OK
 91 ASN1         A32  6.2.0.0        INIT_OK
 96 MBK          A32  6.2.0.0        INIT_OK
 9c ECDSA        A32  6.2.0.0        INIT_OK
 9f CRYPT        A32  6.2.0.0        INIT_OK
```

This command must be executed for the port of the cHSM, not the host. The default port is 4000 + cHSM number.

4. If there is any VDM-using module, uninstall it.

```
csadm Dev=PORT@IP logonSign=ADMIN,:cs2:auto:USB0 DeleteFile=oscca.msc Restart
```

5. Go to the folder containing the firmware modules and install them.

```
cd linux\firmware\<version>\uta
csadm Dev=PORT@IP logonSign=ADMIN,:cs2:auto:USB0 \
    LoadFile=hbs_uta.mtc LoadFile=ml_uta.mtc LoadFile=pqmi_uta.mtc Restart
```

4.1.2 Installing SecurityServer Software

To install the SecurityServer Software on a Linux system, do the following:

1. Copy the downloaded software to the appropriate location on the OpenSSL Server.
2. Create a utimaco folder in the `/opt` directory and create 2 folders within it: `/opt/utimaco/bin` and `/opt/utimaco/lib`.

›_ Console

```
mkdir -p /opt/utimaco/bin
mkdir -p /opt/utimaco/lib
```

3. Copy the PKCS #11 library file `libcs_pkcs11_R3.so` from the Utimaco SecurityServer software to the `/opt/utimaco/lib` directory.

›_ Console

```
cp ./u.trust_anchor_product_bundle-x.x.x/Software/Linux/Crypto_APIs/PKCS11_R3/
lib/libcs_pkcs11_R3.so /opt/utimaco/lib
```

4. Copy the `p11tool2` and `qptool2` files from the u.tyour product bundle and the Utimaco QuantumProtect software to the `/opt/utimaco/bin` directory and make the files executable.

›_ Console

```
cd .\QuantumProtect-x.x.x.x\linux\Crypto_APIs\PKCS11_R3\samples\qptool2\bin
cp qptool2 /opt/utimaco/bin
cd ./u.trust_anchor_product_bundle-x.x.x/Software/Linux/Administration
cp p11tool2 /opt/utimaco/bin
chmod +x /opt/utimaco/bin/qptool2 /opt/utimaco/bin/p11tool2
```

4.2 Setting Up OpenSSL

4.2.1 Installing OpenSSL

If you use an existing or pre-installed OpenSSL software, skip the first two steps.

1. Install the necessary dependencies.

›_ Console

```
dnf -y update && \
    dnf install -y make gcc wget tar git autoconf automake libtool \
    perl-IPC-Cmd perl-core perl-Test-Harness perl-Data-Dumper \
    which findutils diffutils openssl-devel
```

2. Install OpenSSL.

›_ Console

```
# From the SO repository
dnf install openssl
# From the official source
cd /tmp
RUN wget https://www.openssl.org/source/openssl-3.5.0.tar.gz && \
    tar -xzvf openssl-3.5.0.tar.gz && \
    cd openssl-3.5.0 && \
    ./config --prefix=/usr/local/openssl-3.5.0 && \
    make -j$(nproc) && \
    make install && \
    cd ..
```

3. Verify the version of OpenSSL.

›_ Console

```
openssl version
OpenSSL 3.5.0 8 Apr 2025 (Library: OpenSSL 3.5.0 8 Apr 2025)
```

4.2.2 Installing the PKCS#11-Provider

1. Install the PKCS#11-Provider from the official source.

›_ Console

```
cp pkcs11-provider-1.0.tar.gz /tmp/. && \
    cd /tmp && \
    tar -xzvf pkcs11-provider-1.0.tar.gz && \
    cd pkcs11-provider-1.0 && \
    PKG_CONFIG_PATH=/usr/local/openssl-3.5.0/lib64/pkgconfig meson setup
builddir && \
    meson compile -C builddir && \
    meson install -C builddir && \
    cd .. && rm -rf pkcs11-provider-1.0*
```

2. Adjust the environment variables.

›_ Console

```
export PATH="/usr/local/openssl-3.5.0/bin:${PATH}"
export LD_LIBRARY_PATH="/usr/local/openssl-3.5.0/lib64:${LD_LIBRARY_PATH}"
```

The last commands should be added to `/.bashrc` to ensure these environmental variables are set after a reboot.

5 Integration Steps

5.1 Configuration on u.trust Anchor

5.1.1 Configuration of the PKCS#11-Provider

1. Create an `/etc/utimaco` directory.
2. Locate the Utimaco PKCS#11 configuration file in your SecurityServer directory, `./u.trust_anchor_product_bundle-x.x.x/Software/Linux/Crypto_APIs/PKCS11_R3/sample`. Copy the Utimaco PKCS#11 configuration file `cs_pkcs11_R3.cfg` to the `/etc/utimaco` directory.
3. Make the appropriate changes to the `cs_pkcs11_R3.cfg` file.

cs_pkcs11_R3.cfg

```
[Global]
# For unix:
Logpath = /tmp
# Loglevel (0 = NONE; 1 = ERROR; 2 = WARNING; 3 = INFO; 4 = TRACE)
Logging = 1
Keepalive = false
# Set the Device to connect with the HSM
# Device specifier
Device = <PORT@IP>
```

For more information regarding the commands and command parameters, see the Utimaco SecurityServer documentation. The device may be a SecurityServer (PCIe or LAN) device. The device line will follow one of these patterns, based on the HSM form factor:

`Device = 4001@<HSM IP address> Hardware (LAN) HSM`

or

`Device = /dev/cs2.0 Hardware (PCIe) HSM`

To make testing easier, enable the PKCS#11 log file by editing the `Logging Loglevel`. Set `LogPath` and `Logging Loglevel` to `1`. For testing, you can increase them to `4`.

The `LogPath` points to a writable directory, not to a file.

If you encounter problems, check the log file `cs_pkcs11_R3.log` in the directory specified under `LogPath`. During testing, change the `Logging Loglevel` to `1` or `2`. This will limit the logging to only critical messages.

4. Set up the `CS_PKCS11_R3_CFG` environment variable.

```
export CS_PKCS11_R3_CFG=/etc/utimaco/cs_pkcs11_R3.cfg
```

5.1.2 Initializing a PKCS#11-Slot

You must initialize a slot with a custom label using `p11tool2`.

Using the following commands, the SO and user will be initialized.

```
./p11tool2 slot=<slot_no> Label=<token_label> Login=ADMIN,ADMIN.key InitToken=ask
./p11tool2 slot=<slot_no> LoginSO=ask InitPin=ask
```

5.2 Configuration on OpenSSL

5.2.1 Configuring OpenSSL Configuration File

1. Open the `/etc/pki/tls/openssl.cnf` file and enter the following in the first line of the file.

›_ Console

```
openssl_conf = openssl_init
```

2. Enter the following lines under the last section of the `openssl.cnf` file.

›_ openssl.cnf

```
[openssl_init]
providers=provider_sect

[provider_sect]
default = default_sect
pkcs11 = pkcs11_sect

[default_sect]
activate = 1

[pkcs11_sect]
module = /usr/lib/x86_64-linux-gnu/ossl-modules/pkcs11.so
pkcs11-module-path = /opt/utimaco/lib/libcs_pkcs11_R3.so
pkcs11-module-login-behavior = always
activate = 1
```

The `module` and `pkcs11-module-path` variables must be changed according to the user environment.

The variable `module` must point to `pkcs11.so`.

The variable `pkcs11-module-path` must point to the SecurityServer PKCS#11-Provider.

5.2.2 Verify the PKCS#11-Provider

Run the command below to verify if the OpenSSL provider is available.

›_ Console

```
openssl list -providers
Providers:
  default
    name: OpenSSL Default Provider
    version: 3.5.0
    status: active
  pkcs11
    name: PKCS#11 Provider (Utimaco QuantumProtect Fork)
    version: 1.0
    status: active
```

6 Verification and Testing

6.1 Functional Testing

This section describes common cryptographic procedures for testing the integration between the OpenSSL PKCS#11 provider and the Utimaco HSM with Quantum Protect.

6.1.1 Creating a MLDSA key pair

To create a MLDSA key pair, the following command can be used:

```
/opt/utimaco/bin/qptool2 -lib "/opt/utimaco/lib/libcs_pkcs11_R3.so" -s SLOT_ID -p
PIN -token -mldsa -keytype 2 -label KEY_LABEL -gen -count 1
```

To be used by OpenSSL, the private and public parts of the key must share their `CKA_ID`. This can be checked using the p11CAT tool from Utimaco.

2. Verify that the keys are generated on the HSM using the following command:

›_ Console

```
/opt/utimaco/bin/p11tool2 slot=SLOT_ID LoginUser=ask ListObjects
Enter normal user PIN:


CKO_PUBLIC_KEY:

+ 1.1
  CKA_KEY_TYPE                  = CKK_VENDOR_DEFINED
  CKA_UNIQUE_ID                 = C3B21F47-56D3-4433-95C9-41793210D5E8
  CKA_LABEL                     = MLDSA_2_key
  CKA_ID                        = 0x3030 (00)

CKO_PRIVATE_KEY:

+ 2.1
  CKA_KEY_TYPE                  = CKK_VENDOR_DEFINED
  CKA_UNIQUE_ID                 = 056E1AC8-E31A-464F-9336-95C220287A97
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     = MLDSA_2_key
  CKA_ID                        = 0x3030 (00)
```

The PKCS#11 PIN can be directly introduced in the `LoginUser` parameter, but this will be stored in plain text in the command history.

6.1.2 Generating a Certificate from an Existing Key

1. Obtain the key label using p11tool2.

›_ Console

```
p11tool2 slot=SLOT_NUMBER LoginUser=ask ListObjects
```

2. Generate a Certificate Signing Request (CSR).

›_ Console

```
openssl req -new -key "pkcs11:token=<token_label>;object=<key_label>" -out
MLDSA_CSR.csr
```

Here, `key_label` is the key label on the HSM. Provide the Cryptouser PIN and certificate details when prompted.

3. Create the self-signed certificate based on the generated key.

›_ Console

```
openssl req -new -x509 -days 365 -key
"pkcs11:token=<token_label>;object=<key_label>" -out MLDSA.cert
```

6.1.3 Signing and Verifying a Message

1. Create a sample text file with any content inside it.

›_ Console

```
echo "Sample message" > message.txt
```

2. Sign the message file.

›_ Console

```
openssl pkeyutl -sign -in message.txt  -inkey
"pkcs11:token=<token_label>;object=<key_label>" -out signature.sig
```

The result will be the signature of the message in binary format.

3. Verify the signature of the message.

›_ Console

```
openssl pkeyutl -verify -in message.txt -certin -inkey mldsa.cert -sigfile
signature.sig
Signature Verified Successfully
```

The best approach to sign a message would be to use the `openssl cms` command but, unfortunately, this command is still not compatible with PQC algorithms like MLDSA.

6.1.4 Creating a Local CA

1. Open the `/<OPENSSLDIR>/openssl.cnf` file in the text editor and edit the `[CA_default]` section to following:

openssl.cnf

```
dir = /localCA
new_certs_dir = $dir/newcerts
```

You can change `dir` to the directory of your choice, but make sure to use the correct path in the subsequent steps. Here, we have created directory `/localCA` under the root directory, and `new_certs_dir = $dir/newcerts`.

2. Create the directory `/localCA/newcerts`.

›_ Console

```
mkdir /localCA/newcerts
```

3. Create the text files `/localCA/index.txt` and `/localCA/serial`.

›_ Console

```
touch /localCA/index.txt
echo "01" > /localCA/serial
```

4. Create a key pair by using `qptool2` for root CA.

›_ Console

```
./bin/qptool2 -lib "/opt/utimaco/lib/libcs_pkcs11_R3.so" -s SLOT_ID -p PIN
-token -mldsa -keytype 2 -label KEY_LABEL -gen -count 1
```

5. Verify that the keys are generated onto the HSM using the following command:

›_ Console

```
/opt/utimaco/bin/p11tool2 slot=SLOT_ID LoginUser=ask ListObjects
Enter normal user PIN:
CKO_PUBLIC_KEY:
+ 1.1
  CKA_KEY_TYPE                    = CKK_VENDOR_DEFINED
  CKA_UNIQUE_ID                   = C3B21F47-56D3-4433-95C9-41793210D5E8
  CKA_LABEL                       = MLDSA_2_key
  CKA_ID                          = 0x3030 (00)
CKO_PRIVATE_KEY:
+ 2.1
  CKA_KEY_TYPE                    = CKK_VENDOR_DEFINED
  CKA_UNIQUE_ID                   = 056E1AC8-E31A-464F-9336-95C220287A97
  CKA_SENSITIVE                   = CK_TRUE
  CKA_EXTRACTABLE                 = CK_FALSE
  CKA_LABEL                       = MLDSA_2_key
  CKA_ID                          = 0x3030 (00)
```

6. Create the CA certificate based on the generated key that is used for signing other certificates by running the below command.

›_ Console

```
openssl req -new -x509 -days 365 -key
"pkcs11:token=TOKEN_NAME;object=KEY_LABEL;type=private" -out /localCA/newcerts/
ca.cer
```

Here, `CAKey` is the `Object label` for the CA private key on the Utimaco HSM created in Step 5 and, `<token_name>` is the token label. Provide CryptoUser PIN when prompted.

7. A certificate request can be signed by the CA using the following command.

›_ Console

```
openssl ca -policy policy_anything -cert /localCA/newcerts/ca.cer -in /localCA/
newcerts/request.csr -keyfile "pkcs11:token=<token_label>;object=<key_label>"
 -out /localCA/newcerts/certificate.crt
```

7 Troubleshooting

7.1 Common issues and how to resolve them

Error	Diagnosis

`The private key was not found at: pkcs11:token=FedoraCert;object= CertKey1`
 	
`PKCS11_get_private_key returned NULL Could not read private key from org.openssl.engine:pkcs11:pkcs11:token=FedoraCert;object= CertKey1`
	Check that the Key exists on the slot and provide the correct key name.

`open_plugin| W: HSM::ConnectionException(Error::NO_DEVICE_AVAILABLE = 0xbe000007) thrown in select_device Error::NO_DEVICE_AVAILABLE`	Make the changes in the key storage section in the cs_pkcs11.cfg file and check that the HSM device is running and reachable from the host.
`20.06.2025 07:56:39: C_OpenSession returned Error 0x00000005 (CKR_GENERAL_ERROR)`	Check the PKCS#11 log (`/tmp/cs_pkcs11_R3.log` by default) for errors. If it doesn't exist or doesn't get updated, check if the `CS_PKCS11_R3_CFG` environment variable is set with the location of the PKCS#11 configuration file.
`Error occurred on device 4002@IP: Error B0680071 CryptoServer module CXI VDM mech not handled`	The OSCCA module is still installed, and it's interfering with Quantum Protect. Please uninstall the OSCCA module
`LoginUser= failed: p11_login: C_Login [type=1] returned Error 0x00000102 (CKR_USER_PIN_NOT_INITIALIZED)`	PKCS#11 Slot is not initialized. Refer to Initialize a Slot.

Error	Diagnosis

`The CryptoServer PKCS#11 Library R3 is not initialized.`
 	
`Error CKR_CRYPTOKI_NOT_INITIALIZED occurred`
	PKCS#11 Slot is not initialized. Refer to Initialize a Slot.

Table 6: List of errors and their diagnoses

7.2 Log locations and interpretation

The following logs can be reviewed to check for errors:

- Utimaco PKCS#11 provider log: `/tmp/cs_pkcs11_R3.log` by default, configured in the PKCS#11 configuration file.
    - This log contains all the PKCS#11 function calls and attributes.

Error	Diagnosis
`20.06.2025 07:51:25.108 | [00000332:00000332] C_OpenSession| E: HSM::ConnectionException(Error::NO_CONNECTION = 0xbe000015) thrown in open_session Error::NO_CONNECTION`	The provider couldn't connect to the HSM. Please check the PKCS#11 configuration file and ensure the Device section is correctly configured. See Configuration of the PKCS#11 Provider for details.
`20.06.2025 07:10:30.866 | [00000059:00000059] C_GetSlotInfo | E: Error CKR_DEVICE_REMOVED occurred.`	The device was disconnected during the cryptographic operation. This probably happens because of connection issues.

Table 7: List of Common Errors on the PKCS#11 Log File

8 Contact and Support Information

You can reach us from Monday to Friday, 09.00 a.m. to 05.00 p.m., Central European Time (CET).

Utimaco IS GmbH
Germanusstr. 4
52080 Aachen
Germany

RMA Query

If you need to send the device back to Utimaco IS GmbH, please open a new RMA case (Return Merchandise Authorization). We request that you use the following web address. RMA cases cannot be opened by email or phone.

https://support.hsm.utimaco.com/support/rma/new

Other Support Queries

- Mail (preferred contact method)
  support@utimaco.com
  Attach the diagnostic information to your email.
- Web portal
  https://support.hsm.utimaco.com/support/cases/new/
  The diagnostic information will be requested in our response if necessary.
- By phone
  AMERICAS +1-844-UTIMACO (+1 844-884-6226)
  EMEA +49 800-627-3081
  APAC +81 800-919-1301
  The diagnostic information will be requested in our response if necessary.

9 Appendices

9.1 References

Reference	Title	Document Number
[CSADM]	u.trust Anchor – csadm Manual / Utimaco IS GmbH	2021-0037
[UTAADMIN]	u.trust Anchor - Administration Manual / Utimaco IS GmbH	2020-0035
[UTAP11Tool2]	u.trust Anchor - PKCS#11 p11tool2 Reference Manual / Utimaco IS GmbH	2021-0072

Table 8: References

9.2 Command Summary

Task	Command
Create an MLDSA key pair	`/opt/utimaco/bin/qptool2 -lib "/opt/utimaco/lib/libcs_pkcs11_R3.so" -s SLOT_ID -p PIN -token -mldsa -keytype 2 -label KEY_LABEL -gen -count 1`
Generate a Certificate Signing Request from a key	`openssl req -new -key "pkcs11:token=<token_label>;object=<key_label>" -out MLDSA_CSR.csr`
Generate a certificate from a Certificate Signing Request	`openssl req -new -x509 -days 365 -key "pkcs11:token=$PKCS11_TOKEN;object=$KEY_LABEL;pin-value=$PKCS11_PIN" -out test.cert`

Task	Command
Generate a signature from a file using a private key	`openssl pkeyutl -sign -in message.txt -inkey "pkcs11:token=<token_label>;object=<key_label>" -out signature.sig`
Verify a signature using a certificate	`openssl pkeyutl -verify -in message.txt -certin -inkey mldsa.cert -sigfile signature.sig`
Generate a CA from a private key	`openssl req -new -x509 -days 365 -key "pkcs11:token=$PKCS11_TOKEN;object=$KEY_LABEL;pin-value=$PKCS11_PIN" -out ca.cer`
Sign a Certificate Signing Request using a CA private key	`openssl ca -batch -policy policy_anything -cert ca.cer -in request.csr -keyfile "pkcs11:token=$PKCS11_TOKEN;object=$KEY_LABEL" -out certificate.crt`

Table 9: List of Commands