# Paraview

# API Security Gateway

v7.0

# Integration Guide

# u.trust GP HSM

Security Server 6.3.0

utimaco®

## Imprint


| | |
|---|---|
| Copyright 2026 | Utimaco IS GmbH |
| | Krefelder Straße 220 |
| | 52070 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 | 2026-08-31 |
| Status | PUBLISHED |
| Document No. | IG-2026-0095 |

# Table of Contents

# 1 About This Guide

This guide describes how to integrate Paraview API Security Gateway with the Utimaco u.trust GP HSM so that:

- The private key of the server certificate used by the API Security Gateway's HTTPS services is generated, stored and used for handshake signing inside the hardware security boundary of the HSM.
- The post-quantum signature algorithm ML-DSA-65 (FIPS 204) can be used to address the long-term threat that quantum computing poses to conventional RSA/ECDSA certificate hierarchies.

The guide assumes that the Paraview API Security Gateway is already deployed and running. It covers only what must be done from that point onwards to bring the HSM and its post-quantum capability into service.

Two key-custody paths are available and are selected per certificate:

| Path | Where the private key lives | When to use it |
|---|---|---|
| Software key (key_source=local) | Generated by the operator, held encrypted by the product, decrypted into gateway memory only | The recommended way to first validate post-quantum HTTPS in a new environment |
| HSM-managed (key_source=utimaco_hsm) | Generated inside the HSM and never leaves it; signing performed by the HSM through PKCS#11 | Production deployments requiring hardware key custody. Note the deployment prerequisite in *Known Limitations and Considerations* |

Table 1: Key-custody paths

## 1.1 Target Audience

This guide is intended for platform administrators / security officers of the Paraview API Security Gateway and Utimaco HSM administrators. Readers are expected to be familiar with Linux administration, PKI and certificate management, and the basic concepts of PKCS#11.

## 1.2 Purpose of the Integration

- The private key never touches disk: On the HSM-managed path the server certificate private key is generated inside the HSM and is never written to disk or to a configuration file in clear text.
- Quantum resistance: Certificates are signed with ML-DSA-65 as standardized in FIPS 204 and, combined with TLS 1.3 hybrid key agreement (X25519MLKEM768), make the transport layer postquantum ready.
- Centralized management: HSM connection settings and certificates are maintained once in the management console and delivered automatically to every gateway node, so nodes need not be configured individually.

## 1.3 Abbreviations

| Abbreviation | Meaning |
|---|---|
| AES | Advanced Encryption Standard |
| CKM | Cryptoki Mechanism, a PKCS#11 mechanism identifier |
| CXI | Utimaco Cryptographic eXtended Interface |
| GCM | Galois/Counter Mode |
| HBS | Hash-Based Signature (LMS/XMSS) |
| HSM | Hardware Security Module |
| MBK | Master Backup Key |
| ML-DSA | Module-Lattice-Based Digital Signature Algorithm (FIPS 204, formerly CRYSTALS-Dilithium) |

| Abbreviation | Meaning |
|---|---|
| ML-KEM | Module-Lattice-Based Key-Encapsulation Mechanism (FIPS 203, formerly CRYSTALS-Kyber) |
| PCIe | PCI Express |
| PIN | Personal Identification Number |
| PKCS#11 | Public-Key Cryptography Standard #11, cryptographic token interface standard |
| PQC | Post-Quantum Cryptography |
| PQMI | Utimaco post-quantum mechanism distribution firmware module |
| SNI | Server Name Indication |
| SO | Security Officer |
| VDM | Vendor Defined Mechanism |
| GUI | Graphical User Interface |
| TLS | Transport Layer Security |
| API | Application Programming Interface |
| HTTPS | Hypertext Transfer Protocol Secure |

Table 2: Abbreviations

## 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 | Click **Test Connection** |
| `Monospaced` | Code that is given for explanation or as an example, file paths | Edit `cs_pkcs11_R3.cfg` |
| *Italic* | References and important terms | See the *Utimaco p11tool2 Manual* |
| <angle brackets> | Placeholder to be replaced with an actual value | <HSM-IP> |

Table 3: Document conventions

We use special icons to highlight the most important notes and information.

Here you will find important safety information that should be followed.

Here you will find additional notes or supplementary information.

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

# 2 Product Overview

## 2.1 Overview of Paraview API Security Gateway

Paraview API Security Gateway is an enterprise API management and security platform that provides a centralized gateway for securing, managing, and controlling access to APIs and backend services. The platform enables secure HTTPS communication, policy enforcement, authentication and authorization, certificate management, traffic routing, and monitoring capabilities. Through its centralized management console, administrators can configure security settings and distribute configurations across gateway nodes, allowing organizations to manage API security consistently while ensuring secure and reliable API communications across distributed environments.

## 2.2 Overview of Utimaco u.trust GP HSM

Utimaco u.trust GP HSM is a hardware security module developed by Utimaco IS GmbH. It is a physically protected, specialized computer unit designed to perform sensitive cryptographic tasks and securely manage and store cryptographic keys and data. It can be used as a universal, independent security component for heterogeneous computer systems.

With the Quantum Protect firmware loaded, the u.trust GP HSM provides post-quantum algorithms through three firmware modules:

| Firmware module | Capability |
|---|---|
| a5 ML | FIPS 204 ML-DSA / FIPS 203 ML-KEM |
| a6 PQMI | Post-quantum mechanism distribution |
| a2 HBS | Hash-based signatures (LMS / XMSS) |

Table 4: Firmware modules

This integration uses ML-DSA-65 as provided by the a5 ML module.

## 2.3 Joint Value Proposition

The integration of Paraview API Security Gateway with the Utimaco u.trust GP HSM combines centralized API security and management with hardware-backed cryptographic key protection. By securing TLS private keys within the HSM, organizations can protect sensitive cryptographic assets from unauthorized access, strengthen the security of API communications, simplify certificate and key lifecycle management, and support compliance with security and regulatory requirements while maintaining reliable and scalable API services.

This integration replaces the origin of the server certificate's private key, and the algorithm it uses, with the HSM and its post-quantum firmware:

- HSM connection settings are entered once in the management console and delivered to the gateway nodes that need them.
- Certificates are created in the management console — either from an externally generated postquantum key pair, or generated directly inside the HSM — and are delivered to the gateway nodes bound to the same cluster.
- The PKCS#11 client library required to talk to the HSM is either pre-installed on each node or uploaded once and delivered automatically.

Delivery happens over the product's own mutually authenticated configuration channel and takes effect without restarting a gateway node.

# 3 Integration Requirements and Prerequisites

## 3.1 Tested Versions

| Operating System | Paraview API Security Gateway | Utimaco SecurityServer Version | Quantum Protect | Utimaco HSM |
|---|---|---|---|---|
| Linux x86-64 | v7.0 | 6.3.0.0 | 1.5.0.0 ( a5 ML / a6 PQMI / a2 HBS all INIT_OK ) | CryptoServer CSe-Series / Se-Series (LAN or PCIe), or the bl_sim5 simulator for evaluation |

Table 5: Tested versions

The procedures in this guide are based on the standard PKCS#11 interface. The table above lists the tested combination; for physical LAN/PCIe models, deploy the corresponding firmware version.

## 3.2 Hardware and Software Requirements

| Hardware | Hardware Requirements |
|---|---|
| Utimaco LAN HSM | u.trust GP HSM LAN, SecurityServer 6.3.0.0 or later, with Quantum Protect 1.5.0.0 firmware loaded |
| Utimaco PCIe HSM | u.trust GP HSM PCIe, SecurityServer 6.3.0.0 or later, with Quantum Protect 1.5.0.0 firmware loaded |
| Network | The gateway nodes and the management console node can both route to the HSM service port |

Table 6: List of hardware requirements

| Software | Software Requirements |
|---|---|
| System OpenSSL | 3.5 or later on every gateway node (native ML-DSA/ML-KEM support). This is a hard prerequisite for post-quantum capability |
| Product Image | Must be the HSM-enabled build. HSM support is an optional capability of the product image; confirm with your delivery contact that the deployed image includes it — otherwise the HSM operations in this guide report an explicit "HSM support not available" result |
| Utimaco PKCS#11 Library | libcs_pkcs11_R3.so , from the u.trust GP HSM Product Bundle / SecurityServer software package. |
| Utimaco Management Tools | csadm (device management) and p11tool2 (PKCS#11 slot and PIN management) |
| Probing Tool (Optional) | OpenSC pkcs11-tool , for a quick check of slots and tokens |

Table 7: List of software requirements

## 3.3 Prerequisites

Before proceeding with the integration, ensure that the following prerequisites are met:

- The product is deployed and running normally, and you can sign in to the management console as a security officer.
- The Utimaco CryptoServer HSM is deployed and basic configuration is complete (see the official CryptoServer documentation).
- An MBK has been created and safely stored for every HSM.
- The default CryptoServer administrator has been replaced with a newly created administrator user.
- Quantum Protect firmware is loaded and a5 ML , a6 PQMI and a2 HBS are all INIT_OK (verification in *Confirming the Post Quantum Firmware Modules* ).

- The system OpenSSL on every gateway node is 3.5 or later (in domestic-stack or offline environments, verify the version in the base image).
- The product's encryption key ( STOA_ENCRYPTION_KEY , 32 bytes base64) is configured, and every gateway node uses the same value as the management console (see *Encryption Key Consistency*).
- You have the administrative privileges required to install software on the nodes.
- You have download permissions on the Utimaco support portal: https://support.hsm.utimaco.com/

# 4 Configuring Utimaco SecurityServer

Goal of this chapter: Bring the HSM into a ready state and provide a Cryptographic User that the product can log in with. Keys are not created here — on the HSM-managed path they are generated inside the HSM when triggered from the management console in *HSM-Managed-ML-DSA-65*.

Install the SecurityServer software as described in the SecurityServer Manual; we recommend uninstalling any earlier version first. `libcs_pkcs11_R3.so` is found under `Software/Linux/Crypto_APIs/PKCS11_R3/lib/` in the product package — it belongs to the base software package, not to the Quantum Protect add-on.

## 4.1 Confirming the Post-Quantum Firmware Modules

```
# <DEVICE> is <port>@<HSM-IP> (LAN), /dev/cs2.0 (PCIe), or 3001@127.0.0.1
(simulator) csadm dev=<DEVICE> listfirmware
```

All three modules must report INIT_OK:

```
a2 HBS ... INIT_OK
a5 ML ... INIT_OK ← FIPS 204 ML-DSA / FIPS 203 ML-KEM
a6 PQMI ... INIT_OK ← post-quantum mechanism distribution
```

If the post-quantum modules are missing on a physical device, the HSM administrator must load the Quantum Protect firmware before you continue: csadm dev=<DEVICE> LogonSign <ADMIN>, <ADMIN.key> LoadFile=hbs.mtc LoadFile=ml.mtc LoadFile=pqmi.mtc Restart

The bl_sim5 simulator in the Quantum Protect evaluation package already contains the PQC firmware. That binary is 32-bit i386, so on a 64-bit host install the 32-bit runtime first ( libc6:i386 , libstdc++6:i386 , libgcc-s1:i386 ), then start it with ./cs_sim.sh & rom\ linux/sim5_linux/bin — it listens on 3001@127.0.0.1 by default.

## 4.2 Configuring cs_pkcs11_R3.cfg

The Utimaco PKCS#11 library is not self-contained: once loaded, it still needs a configuration file giving the HSM device address, and the environment variable `CS_PKCS11_R3_CFG` must point to that file.

Place `cs_pkcs11_R3.cfg` on every node whose processes access the HSM — the management console node and each gateway node, or inside the deployment image.

```
[Global]
Logging = 0 ; temporarily set to 4 when troubleshooting
LogFile = /var/log/stoa/cs_pkcs11_R3.log
[CryptoServer]
Device = <DEVICE> ; LAN: <port>@<HSM-IP>; PCIe: /dev/cs2.0
ConnectionTimeout = 5000
CommandTimeout = 60000
```

- LAN models: Device = <port>@<HSM-IP> (for example 288@10.10.x.x ). In containerized deployments the container network must be able to route to the HSM.
- PCIe cards: Device = /dev/cs2.0 . In containerized deployments the device must be passed through into the container.

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

Deployment on the HSM device side — firmware initialization, exposure as a network service, clustering and high availability — is performed by HSM operations according to the on-site architecture and the official Utimaco documentation, and is out of scope for this guide.

## 4.3 Creating a Cryptographic User and Initializing the Slot

Critical privilege requirement: ML-DSA key generation requires a Cryptographic User (privilege bit 0x02 ) whose CXI_GROUP covers the group holding the key ( SLOT_0000 by default). Logging in as ADMIN succeeds, but key generation then fails with 0xB0A60001 (PQMI permission denied).

Use p11tool2 with the ADMIN key to initialize slot 0. This also creates the security officer SO_0000 and the cryptographic user USR_0000 ( CXI_GROUP=SLOT_0000 ).

```
export CS_PKCS11_R3_CFG=/etc/stoa/cs_pkcs11_R3.cfg
# 1) Initialize the token as administrator, setting an initial SO PIN
p11tool2 Slot=0 Login=ADMIN,<ADMIN.key> InitToken=<initial PIN>
# 2) Change the initial SO PIN to the formal SO PIN
p11tool2 Slot=0 LoginSO=<initial PIN> SetPIN=<initial PIN>,<SO-PIN>
# 3) Initialize the user PIN as SO
p11tool2 Slot=0 LoginSO=<SO-PIN> InitPIN=<initial user PIN>
# 4) Change the initial user PIN to the formal user PIN
p11tool2 Slot=0 LoginUser=<initial user PIN> SetPIN=<initial user PIN>,<user PIN>

# Verify: USER_PIN_INITIALIZED must be CK_TRUE
p11tool2 Slot=0 LoginUser=<user PIN> GetTokenInfo

# Verify privileges and group: USR_0000 must show 00000002 and group SLOT_0000
csadm dev=<DEVICE> LogonSign=ADMIN,<ADMIN.key> ListUsers
# USR_0000 00000002 HMAC passwd ...A[CXI_GROUP=SLOT_0000]
```

InitToken and InitPIN set only an initial PIN. You must run SetPIN again to change it to a formal PIN, otherwise firmware policy may cause later operations to fail with CKR_PIN_TOO_WEAK . Note: The user PIN length must be between 8 and 255 characters. This <user PIN> is the value you later enter as pin in the HSM connection settings (Section 6.1). Security note: All PINs above are placeholders. In production, set strong PINs and store them only encrypted, through the management console; never write them in clear text into configuration files or logs. We recommend the ask interactive mode of p11tool2 so PINs do not enter the shell history.

## 4.4 Checking the Slot and Token

```
CS_PKCS11_R3_CFG=/etc/stoa/cs_pkcs11_R3.cfg \
pkcs11-tool --module /opt/utimaco/lib/libcs_pkcs11_R3.so -L
```

The output should show slot 0 with token label CryptoServer PKCS11 Token and manufacturer Utimaco IS GmbH.

The slot_label you enter in the HSM connection settings must be the token's CKA label ( CryptoServer PKCS11 Token by default), not the slot description SLOT_0000. Slots are matched by token label; an incorrect value produces the error "no token slot found with label X".

# 5 Configuring Paraview API Security Gateway

## 5.1 System OpenSSL: A Hard Prerequisite

Post-quantum TLS on a gateway node depends on the system OpenSSL. On every gateway node:

```
openssl version # expect OpenSSL 3.5.x or later
```

OpenSSL earlier than 3.5 does not support ML-DSA. Classic RSA/ECDSA certificates are unaffected, but post-quantum ML-DSA certificates require system OpenSSL 3.5 or later on the node. In domestic-stack or offline environments, verify the OpenSSL version of the base image and ship it with the deployment package if necessary.

The client side likewise needs OpenSSL 3.5 or later (or another ML-DSA implementation) to complete a post-quantum handshake. See *Verifying the Post Quantum TLS Handshake*.

## 5.2 Placing the PKCS#11 Client Library

Required only for the HSM-managed path. Skip for the software key path.

Option 1 — pre-install on the node (common for domestic-stack and offline images): place `libcs_pkcs11_R3.so` and `cs_pkcs11_R3.cfg` on the node or in the image, and set `pkcs11_module_path` in the HSM connection settings to the absolute path of that .so, for example `/opt/utimaco/lib/libcs_pkcs11_R3.so`.

Option 2 — upload once and let the product deliver it (recommended for multiple nodes): upload the .so through the management console (*Delivering PKCS11 Client Library*). Each gateway node then downloads it, verifies its SHA-256 and writes it atomically to `pkcs11_module_path`, with no per-node manual work.

With either option, `cs_pkcs11_R3.cfg` and the CS_PKCS11_R3_CFG environment variable must be provided by the node side (pre-installed or injected by the deployment system). Delivery covers only the .so itself; see item 2 in *Known Limitations and Considerations*.

Client library delivery currently assumes the target architecture linux-x86-64.

## 5.3 Enabling the HTTPS Listener

In the gateway node configuration file (stoa.yaml), make sure the HTTPS listener is enabled and the node is bound to the right cluster.

```
gateway:
listen_addr: "0.0.0.0:6188" # plaintext port
tls_listen_addr: "0.0.0.0:8443" # HTTPS port — setting this enables dynamic-
certificate T
LS
control_plane:
cluster_id: "<UUID of the cluster this node belongs to>"
node_name: "<node name>"
```

- Setting tls_listen_addr enables HTTPS; certificates may arrive afterwards. While no certificate is present, handshakes for that SNI fail, but the node does not crash and the plaintext port is unaffected.
- Certificates are hot-applied: once delivered, new connections use them immediately and in-flight connections are not interrupted. Changing tls_listen_addr itself requires restarting the node.
- cluster_id determines which certificates and HSM settings the node receives — it must match the cluster you bind certificates to in *Issuing Post-Quantum ML-DSA-65 Certificates*.

## 5.4 Encryption Key Consistency (Critical)

Software private keys and HSM PINs are delivered as AES-256-GCM ciphertext keyed by STOA_ENCRYPTION_KEY. Every gateway node must be configured with exactly the same value as the management console:

```
export STOA_ENCRYPTION_KEY="<exactly the same 32-byte base64 value as the
management console>"
```

If the value is missing or different, software-key certificates fail to decrypt and are skipped (a warning is logged) and HTTPS for those certificates is unavailable. The PIN of HSM-source certificates also cannot be decrypted, so HSM login fails.

The deployment system should supply this key to all gateway nodes over a secure channel. Do not write it in clear text into the configuration file.

## 5.5 Ports

| Port | Purpose | How to Override |
|---|---|---|
| 6188 | Plaintext service port. | gateway.listen_addr |
| 8443 | HTTPS service port. | gateway.tls_listen_addr |
| 9095 | Node administrative endpoint, including the HSM connectivity test. | STOA_ADMIN_PORT |

Table 8: Ports

# 6 Integration Steps on Paraview API Security Gateway

The operations in this chapter require the security:manage_certificates capability (security officer role). Examples use curl; exact fields depend on the API version deployed on site.

## 6.1 Entering the HSM Connection Settings

UI path: **Cluster Certificate Management** → **HSM Integration** → **New Configuration**. The PIN is never displayed.

```
curl -X POST http://<console>:8081/api/v1/hsm-configs \
-H "Authorization: Bearer <token>" -H 'Content-Type: application/json' \
-d '{
    "name": "utimaco-prod-1",
    "vendor": "utimaco",
    "pkcs11_module_path": "/opt/utimaco/lib/libcs_pkcs11_R3.so",
    "slot_label": "CryptoServer PKCS11 Token",
    "pin": "<user PIN>",
    "key_label_prefix": "stoa-",
    "enabled": true,
    "cluster_id": "<target cluster UUID, optional>"
}'
```

Key fields (full list in Appendix B):

- pkcs11_module_path — absolute path of the vendor library on the gateway node; also the target path when the product delivers the library.
- slot_label — must be the token's CKA label ( CryptoServer PKCS11 Token by default), not the slot description SLOT_0000.
- pin — the user PIN from Section 4.3. It is submitted once and immediately encrypted with AES 256-GCM; it is never stored in clear text and never appears in API responses, logs or the UI.
- cluster_id — the gateway cluster to bind to. Leaving it empty makes the configuration a global fallback, used by any cluster without a dedicated binding.
- enabled — only enabled configurations are delivered.

Post-quantum mechanism values. For the HSM-managed path, the management console node also needs the PKCS#11 mechanism value for ML-DSA key generation:

```
export CS_PKCS11_R3_CFG=/etc/stoa/cs_pkcs11_R3.cfg
export STOA_HSM_CKM_ML_DSA_65=0xFFFFFFFFC0A50001 # must be 64-bit sign-extended
```

The mechanism value must use the 64-bit sign-extended form. Utimaco's MECH_ML_VDM = (0xC<<28)|(0xA5<<16) = 0xC0A50000 is negative as an int32 and is signextended when converted to CK_ULONG . The correct values are 0xFFFFFFFFC0A50001 (key generation) and 0xFFFFFFFFC0A53001 (signing). The 32-bit value 0xC0A50001 results in CKR_GENERAL_ERROR . Important (decisive pitfall 2): The VDM type of the a5 ML module uses the enumerated values 44→1, 65→2, 87→3 — not the PQMI/CXI values 0x32/0x33/0x35 . ML-DSA-65 must be passed as 2; passing 0x33 yields firmware error 0xB0A5000E .

## 6.2 Testing the Connection

UI path: **Configuration card → Test Connection**.

```
curl -X POST http://<console>:8081/api/v1/hsm-configs/<id>/test \
-H "Authorization: Bearer <token>"
# On success: {"ok":true,"message":"...","slot_info":{...}}
```

The test performs the full sequence — load the library, initialize it, find the slot by token label, log in, read token information. A failure at any step returns a reason precise enough to locate the problem.

## 6.3 Delivering the PKCS#11 Client Library

Required only when using option 2 of Section *Placing the PKCS#11 Client Library*. Skip if the library is pre-installed on the nodes.

```
curl -X POST http://<console>:8081/api/v1/hsm-client-libs \
-H "Authorization: Bearer <token>" \
-F "file=@libcs_pkcs11_R3.so" -F "vendor=utimaco" -F "arch=linux-x86-64"
# Maximum 64 MB per file; the server computes the SHA-256 and deduplicates by
digest
```

Then associate the uploaded artifact with the HSM configuration ( client_lib_id ). Each gateway node fetches it, verifies the SHA-256 and writes the file atomically to pkcs11_module_path .

# 7 Issuing Post-Quantum ML-DSA-65 Certificates

## 7.1 Path A: Software Key ML-DSA-65 (Recommended First)

Generate the key and certificate with OpenSSL 3.5 (*Generating a Test ML-DSA-65 Certificate*), then register them:

```
curl -X POST http://<console>:8081/api/v1/certificates \
-H "Authorization: Bearer <token>" -H 'Content-Type: application/json' \
-d '{
    "name": "gw-mldsa",
    "domain": "gw.example.com",
    "cert_pem": "<ML-DSA-65 certificate chain in PEM>",
    "key_pem": "<ML-DSA-65 private key in PEM>",
    "cert_type": "external_ca",
    "key_source": "local",
    "key_algorithm": "ml-dsa-65",
    "cluster_id": "<target cluster UUID>"
}'
```

The private key is stored encrypted, delivered as ciphertext, and decrypted by the gateway node into memory only.

## 7.2 Path B: HSM-Managed ML-DSA-65

UI path: **Cluster Certificate Management** → **Add Certificate** → **key source Utimaco HSM managed** → **key algorithm Post-quantum ML-DSA-65** (the UI notes that Quantum Protect firmware is required).

```
curl -X POST http://<console>:8081/api/v1/certificates \
-H "Authorization: Bearer <token>" -H 'Content-Type: application/json' \
-d '{
  "name": "gw-mldsa-hsm",
  "domain": "gw.example.com",
  "key_source": "utimaco_hsm",
  "key_algorithm": "ml-dsa-65",
  "hsm_key_label": "<key label inside the HSM; defaults to
<key_label_prefix><name>>",
  "cluster_id": "<target cluster UUID>"
}'
```

An ML-DSA-65 key pair is then generated inside the HSM through PKCS#11 — the private key never leaves it — and the certificate records the key source, the algorithm and the key reference. The list view shows it as "HSM managed · ML-DSA-65".

key_source defaults to local . For HSM-managed certificates key_algorithm defaults to ecdsa, so ml-dsa-65 must be specified explicitly to obtain post-quantum. Valid algorithm families are rsa , ecdsa , sm2 and ml-dsa-65.

## 7.3 Activation, Rotation and Revocation

- No manual delivery step is needed. After a certificate is created, rotated, deleted or revoked, it is delivered to the online gateway nodes of its cluster and acknowledged. A node that (re)connects receives the full current set for its cluster_id .
- Hot activation: new connections use the new certificate immediately; a node restart is not required.
- Rotation (POST `/api/v1/certificates/<id>/rotate`) retains the original key source and algorithm by default; both can be overridden explicitly in the request.

# 8 Verification and Testing

This chapter verifies only the Paraview API Security Gateway and the HSM — that the HSM can perform ML-DSA, that the key resides in the HSM, that the certificate is applied, and that a post-quantum handshake is negotiated. It requires no other business system.

## 8.1 Confirming HSM Algorithm Support (Optional but Recommended)

Independently confirm that the HSM can perform ML-DSA-65 key generation, signing and verification, using the Utimaco Java CXI samples from the evaluation package (pure socket, no vendor *.so* required):

```
java -cp "CryptoServerAPI-<version>-linux.jar:<compiled-samples>" \
  com.utimaco.FullTestSuite -mldsa -suite -p <cryptographic user>,<user PIN>
```

Expected (ML-DSA-65 corresponds to keytype 0x33 → MLDSA-...-51-1):

```
MLDSA-PSEU-EXT-51-1 Keytype is 51 (33)
  GEN STAGE: PASSED
  SIGN STAGE: PASSED
  VERIFY STAGE: PASSED
```

Tip (known sample issues): On the single-keytype path, announceMLDSA reports "Invalid MLDSA key type" — use -suite to bypass it. The sample's key-file login fails with 0xb906100e because a two-part credential is treated as an HMAC password; the -p <user>,<PIN> form shown above is password login and works unmodified.

## 8.2 Testing HSM Connectivity from a Gateway Node

```
curl -X POST http://<gateway-node>:9095/v1/admin/hsm/test \
  -H "x-admin-key: <STOA_ADMIN_KEY>"
# On success: {"ok":true,"results":
[{"token_label":"...","manufacturer":"...","slot_id":0}]}
```

For every delivered HSM configuration this performs load → initialize → find slot by label → log in → read token information. If the deployed image does not include HSM support, it returns 501 with an explicit message rather than failing silently.

## 8.3 Verifying the Post-Quantum TLS Handshake

```
# TLS version, post-quantum KEM and post-quantum signature in one line
curl -vk --tlsv1.3 https://<gateway>:8443/ 2>&1 | grep "SSL connection"
# Expect: TLSv1.3 / TLS_AES_256_GCM_SHA384 / X25519MLKEM768 / id-ml-dsa-65

# Negotiation details and certificate subject
echo | openssl s_client -connect <gateway>:8443 -tls1_3 2>/dev/null \
| grep -E "Protocol|Peer signature type|Negotiated TLS1.3 group|subject="

# Certificate signature algorithm
echo | openssl s_client -connect <gateway>:8443 2>/dev/null \
| openssl x509 -noout -text | grep -i "Signature Algorithm" # → ML-DSA-65
```

A self-signed test certificate produces Verify return code: 18 (self-signed), which is expected. In production use a certificate issued by a trusted CA, or validate the chain with -CAfile .Important: A client with OpenSSL earlier than 3.5 reports an unsupported algorithm or version. That is a client limitation, not a server fault.

## 8.4 Generating a Test ML-DSA-65 Certificate

```
openssl req -x509 -new -newkey ml-dsa-65 -noenc \
  -keyout mldsa.key -out mldsa.crt -days 365 \
  -subj "/CN=gw.example.com" -addext "subjectAltName=DNS:gw.example.com"

openssl x509 -in mldsa.crt -noout -text | grep -iE "Signature Algorithm|Public Key
Algorithm"
# Signature Algorithm: ML-DSA-65
# Public Key Algorithm: ML-DSA-65
```

(The OID of ML-DSA-65 is id-ml-dsa-65 = 2.16.840.1.101.3.4.3.18.)

## 8.5 Acceptance Tests

| No. | Verification Item | Method | Expected Result |
|---|---|---|---|
| 1 | HSM post-quantum firmware ready | `csadm listfirmware` | a5 ML, a6 PQMI, a2 HBS all INIT_OK |
| 2 | PKCS#11 library can reach the HSM | `pkcs11-tool -L` | Token CryptoServer PKCS11 Token listed |
| 3 | HSM algorithm support (optional) | *CXI suite* | ML-DSA GEN/ SIGN/VERIFY all PASSED |
| 4 | Console connects to the HSM | *Testing the Connection* | ok:true |
| 5 | Gateway node connects to the HSM | *Testing HSM Connectivity from a Gateway Node* | ok:true |
| 6 | Key resides in the HSM (managed path) | Certificate shows HSM key source with a key reference; the matching label is visible on the HSM | Private key never leaves the HSM |
| 7 | Delivery and hot activation | Start a handshake after creating the certificate | Takes effect without a restart |

| No. | Verification Item | Method | Expected Result |
|---|---|---|---|
| 8 | Post-quantum signature algorithm | *Verifying the Post Quantum TLS Handshake* | Signature Algorithm: ML-DSA-65 |
| 9 | Post-quantum key agreement | *Verifying the Post Quantum TLS Handshake* | X25519MLKEM768 |
| 10 | Business traffic works | Send a request through the HTTPS port | Proxied to the upstream normally |

Table 9: Acceptance Tests

# 9 Go-Live Checklist

- [ ] openssl version is 3.5 or later on every gateway node.
- [ ] The deployed product image is the HSM-enabled build.
- [ ] STOA_ENCRYPTION_KEY is identical on every gateway node and the management console.
- [ ] The HSM is reachable and csadm listfirmware shows a5 ML ... INIT_OK.
- [ ] A Cryptographic User exists (privilege bit 0x02 , group covering the key group) and its initial PIN has been changed to a formal strong PIN.
- [ ] cs_pkcs11_R3.cfg is in place and CS_PKCS11_R3_CFG is set on the console node and every gateway node.
- [ ] pkcs11-tool -L lists the token CryptoServer PKCS11 Token.
- [ ] slot_label equals the token CKA label (not SLOT_0000).
- [ ] The .so is pre-installed or uploaded for delivery, and pkcs11_module_path matches the actual path on disk.
- [ ] STOA_HSM_CKM_ML_DSA_65 uses the 64-bit sign-extended value.
- [ ] Certificates are created and bound to the correct cluster_id.
- [ ] tls_listen_addr is configured on every gateway node.
- [ ] The HSM connection test returns ok:true from both the console and a gateway node.
- [ ] curl -vk --tlsv1.3 shows id-ml-dsa-65 and X25519MLKEM768.
- [ ] The MBK is created and held in split custody according to the organization's key escrow policy.
- [ ] PINs and recovery credentials are stored according to organizational policy and appear in no configuration file or log.

# 10 Known Limitations and Considerations

| # | Item | Description |
|---|---|---|
| 1 | HSM-managed handshake signing has a deployment prerequisite | Using an HSM-resident private key as the TLS handshake key requires the OpenSSL 3.x pkcs11 provider on the node (loading a pkcs11: URI as a key through OSSL_STORE). Without it, the SNI handshake for HSM-source certificates fails and degrades to HTTP 502. The node does not crash, and software-key certificates and other SNIs are unaffected. Prerequisite: install and configure the OpenSSL pkcs11 provider (it must be self-contained in the deployment package for offline environments). Software-key ML-DSA-65 certificates are not subject to this limitation and are fully operational end to end, so validate post-quantum HTTPS on the software key path first in a new environment. |
| 2 | The Utimaco .so is not self-contained | Delivery covers only the .so; cs_pkcs11_R3.cfg and CS_PKCS11_R3_CFG must be provided by the node side, pre-installed or injected by the deployment system. |
| 3 | Single architecture | Client library delivery currently assumes linux-x86-64. |

| # | Item | Description |
|---|---|---|
| 4 | Client compatibility | A post-quantum handshake requires the client to support ML-DSA (OpenSSL 3.5 or later, or equivalent). For public-facing services that must remain compatible with existing clients, also provide classic RSA/ECDSA certificates and let SNI/ negotiation select automatically. |
| 5 | Credential hygiene | All PINs, device addresses and key labels in this guide are placeholders. In production they must be replaced with real strong credentials and stored only encrypted through the management console. |

Table 10: Limitations

# 11 Troubleshooting

## 11.1 Common Issues and How to Resolve Them

| Symptom / Error | Possible Cause | Resolution |
|---|---|---|
| **Handshake fails with "no certificate for SNI"** | Certificate not delivered, incorrect cluster binding, or certificate not yet available on the gateway node. | Verify that the certificate is bound to the correct cluster and that the gateway node has synchronized successfully. |
| **Software-key certificate fails with a "decrypt key" error** | The STOA_ENCRYPTION_KEY value differs between the management console and gateway node. | Ensure that the same STOA_ENCRYPTION_KEY value is configured on all nodes and the management console. |
| **HSM-managed certificate handshake returns HTTP 502** | OpenSSL PKCS#11 provider is not installed or configured. | Install and configure the PKCS#11 provider or use the software-key deployment model. |
| **pkcs11-tool -L cannot connect or no slots are displayed.** | CS_PKCS11_R3_CFG is not set, or the HSM device address is incorrect. | Verify the configuration file location, environment variable, and HSM device address. |
| **Connection test reports "no token slot found with label X"** | Incorrect token label configured. | Configure the token label as CryptoServer PKCS11 Token instead of the slot description (SLOT_0000). |
| **Connection test reports "cannot load PKCS#11 module"** | PKCS#11 library is missing or the configured path is incorrect. | Verify that libcs_pkcs11_R3.so is installed and that the configured module path is correct. |
| **HSM support unavailable (HTTP 501)** | The deployed gateway image does not include HSM support. | Deploy the HSM-enabled build of Paraview API Security Gateway. |

| | | |
|---|---|---|
| **Login succeeds, but key generation fails with 0xB0A60001** | An administrator account was used instead of a Cryptographic User. | Use a Cryptographic User account with the required permissions. |
| **Login fails with 0xb906100e** | Incorrect credential type used (password versus key file). | Use password-based login for user credentials and key-file authentication where applicable. |
| **Operation fails with CKR_PIN_TOO_WEAK** | Initial PIN was not changed after slot initialization. | Change the initial PIN to a strong production PIN. |
| **TLS handshake reports an unsupported algorithm or version.** | OpenSSL version is earlier than 3.5. | Upgrade the client or gateway OpenSSL installation to version 3.5 or later. |

Table 11: Troubleshooting

In-depth troubleshooting: Temporarily set Logging to 4 in cs_pkcs11_R3.cfg . The vendor library then writes every C_* call and every Utimaco::HSM::DeviceException (error_code=…) to the file given by LogFile , which pinpoints the PKCS#11 call at which the failure occurs.

# 12 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
Krefelder Straße 220
52070 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.

# 13 Appendix A: Quick Reference

| Item | Value |
|---|---|
| ML-DSA-65 OID | id-ml-dsa-65 = 2.16.840.1.101.3.4.3.18 |
| ML-DSA key generation mechanism (PKCS#11 CKM) | 0xFFFFFFFFC0A50001 (64-bit sign-extended) |
| ML-DSA signing mechanism (PKCS#11 CKM) | 0xFFFFFFFFC0A53001 |
| a5 ML module VDM type enumeration | 44→1 / 65→2 / 87→3 |
| Key generation mechanism parameter (VDM serialization, big endian) | MLDSA_KEYGEN{ flags(u4)=1, type(u4)=<enum>, attrs(v2), seed(*) } |
| ML-DSA-65 sizes (FIPS 204) | Public key: 1952 B / Private key: 4032 B / Signature: 3309 B |
| Token CKA label (default) | CryptoServer PKCS11 Token |
| Post-quantum firmware modules | a5 ML (FIPS 204/203) / a6 PQMI / a2 HBS |
| Cryptographic User privilege bit | 0x02 |
| User PIN length | 8–255 |
| Minimum system OpenSSL | 3.5 (server and client) |
| Expected TLS negotiation | TLSv1.3 / TLS_AES_256_GCM_SHA384 / X25519MLKEM768 / id-ml-dsa-65 |

| Item | Value |
| --- | --- |
| Environment variables | CS_PKCS11_R3_CFG, STOA_ENCRYPTION_KEY, STOA_HSM_CKM_ML_DSA_65, STOA_ADMIN_KEY |

Table 12: Quick Reference

# 14 Appendix B: API Reference

## 14.1 HSM Connection Settings ( `/api/v1/hsm-configs` )

| Field | Type | Description |
|---|---|---|
| **name** | string, required | Configuration name (1–128 characters). |
| **vendor** | string | Vendor; currently only utimaco. |
| **pkcs11_module_path** | string, required | Absolute path of the PKCS#11 shared library on the gateway node; also the delivery target path. |
| **slot_label** | string | The PKCS#11 token CKA label (CryptoServer PKCS11 Token by default); up to 128 characters. |
| **pin** | string, required | Cryptographic User PIN. Submitted once, then stored AES-256-GCM encrypted; never appears in responses, logs, or the UI. |
| **key_label_prefix** | string | Prefix for key labels inside the HSM (stoa- by default); up to 64 characters. |
| **gm_enabled** | bool | Chinese-national-algorithm firmware capability flag (connection level; signature algorithm is chosen per certificate). |
| **enabled** | bool | Whether the configuration is active; only enabled configurations are delivered. |
| **cluster_id** | UUID | Gateway cluster to bind to; empty means global fallback. |

| | | |
|---|---|---|
| **client_lib_id** | UUID | Referenced PKCS#11 client library artifact. |

Table 13: HSM connection settings

## 14.2 Certificate Key Fields ( `/api/v1/certificates` )

| Field | Values | Description |
|---|---|---|
| key_source | local (default) / utimaco_hsm | Private key source. With utimaco_hsm the key is generated inside the HSM and a key reference is recorded. |
| key_algorithm | rsa / ecdsa / sm2 / ml-dsa-65 | Key algorithm family. Defaults to ecdsa for HSM-managed certificates, so ml-dsa-65 must be given explicitly for post-quantum. With local it is implied by the PEM and may be omitted. |
| hsm_key_label | string | Key label inside the HSM; defaults to <key_label_prefix><name>. |
| key_pem | string | Only for key_source=local; stored and delivered encrypted. |
| cluster_id | UUID | Gateway cluster binding, which determines the delivery scope. |

Table 14: Certificate key fields

## 14.3 PKCS#11 Client Library ( `/api/v1/hsm-client-libs` )

| Field | Description |
|---|---|
| file | Multipart file field, up to 64 MB. |
| filename | File name written on the gateway node (for example, libcs_pkcs11_R3.so). |
| vendor | Currently only Utimaco. |
| arch | Target architecture; currently only linux-x86-64. |
| version | Version marker (optional, up to 64 characters). |
| sha256 | Computed by the server; used for upload idempotency and integrity verification. |

Table 15: PKCS#11 Client Library