Cloudflare

Keyless SSL

1.18.0

Integration Guide

u.trust GP HSM

6.6.0

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	2.0.0
Date	2026-10-19
Status	PUBLISHED
Document No.	IG-2026-0059

All rights reserved	No part of this documentation may be reproduced in any form (printing, photocopy or according to any other process) without the written approval of Utimaco IS GmbH or be processed, reproduced or distributed using electronic systems. Utimaco IS GmbH reserves the right to modify or amend the documentation at any time without prior notice. Utimaco IS GmbH assumes no liability for typographical errors and damages incurred due to them. All trademarks and registered trademarks are the property of their respective owners.

Table of Contents

1 Introduction 5
1.1 About This Guide 5
1.2 Target Audience 5
1.3 Purpose of the Integration 5
1.4 Abbreviations 6
1.5 Document Conventions 8
2 Product Overview 10
2.1 Cloudflare Keyless SSL 10
2.2 Utimaco u.trust GP HSM 10
2.3 Joint Value Proposition 10
3 Integration Requirements and Prerequisites 12
3.1 Tested Versions 12
3.2 Software Requirements 12
3.3 Hardware Requirements 13
3.4 Prerequisites 13
4 Installation and Configuration of Utimaco SecurityServer Software 15
4.1 Downloading and Installing Utimaco Software 15
4.2 u.trust GP HSM PKCS#11 Configuration 15
4.3 u.trust GP HSM Simulator Setup and Initialization 17
4.4 Create SO User and Initialize a Slot 19
4.5 Installation and Configuration of Cloudflare Keyless SSL 20
4.5.1 Installation of GoKeyless 20
4.5.2 Cloudflare Dashboard Configuration 22
5 Integration Steps 24
5.1 Generating Key on Utimaco HSM 24
5.2 CSR Generation Using HSM-Based Private Key 24
5.3 Signing CSR with Public Certificate Authority 25
5.4 Upload the Keyless SSL Certificate 27
5.5 GoKeyless Configuration 29
6 Verification and Testing 31
6.1 Service Startup and Integration Validation 31

6.1.1 Nginx Installation and Configuration 31
6.1.2 GoKeyless Service Setup and Initialization 32
6.2 Logs and Validation Steps 33
7 Troubleshooting 36
7.1 Common issues and how to resolve them 36
7.2 Log locations and interpretation 36
8 Contact and Support Information 37
9 Appendices 38
9.1 Command Summary (CLI commands used) 38

1 Introduction

This guide is part of the support provided by Utimaco. Additional documentation produced to support your Utimaco u.trust General Purpose (GP) Hardware Security Module (HSM) product can be found in the document directory of the Utimaco u.trust GP HSM product bundle.

All Utimaco u.trust GP HSM product documentation is available from Utimaco's website at *https://utimaco.com/*.

1.1 About This Guide

This guide describes how to integrate a Utimaco u.trust GP HSM with Cloudflare Keyless SSL. The Utimaco HSM securely stores the private key used for TLS/SSL and performs all cryptographic signing operations within the hardware, ensuring that the private key never leaves the HSM. Cloudflare Keyless SSL enables secure delivery of web traffic through Cloudflare's global network while allowing the private key to remain under the customer's control.

In this integration, Cloudflare handles the majority of the TLS handshake at the edge. Whenever a private key operation is required, Cloudflare securely forwards the request to a customer-managed key server running the GoKeyless service. This service interacts with the Utimaco HSM via the PKCS#11 interface to perform cryptographic operations such as signing. This approach ensures that sensitive key material is protected within the HSM at all times, while still leveraging Cloudflare's CDN, DDoS protection, and performance optimizations.

1.2 Target Audience

This guide is intended for Cloudflare Keyless SSL and Utimaco HSM administrators.

1.3 Purpose of the Integration

The integration of Cloudflare Keyless SSL with u.trust GP HSM serves a critical role in enhancing the security, compliance, and performance of TLS/SSL operations by ensuring that private keys are securely stored within the HSM while leveraging Cloudflare's global edge network for handling client connections and traffic optimization.

1.4 Abbreviations

Abbreviation	Meaning
HSM	Hardware Security Module
PKCS#11	Public-Key Cryptography Standards #11 (Cryptographic Token Interface)
SSL	Secure Sockets Layer
TLS	Transport Layer Security
CSR	Certificate Signing Request
CA	Certificate Authority
DNS	Domain Name System
FQDN	Fully Qualified Domain Name
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
API	Application Programming Interface
mTLS	Mutual Transport Layer Security
SKI	Subject Key Identifier

Abbreviation	Meaning
PEM	Privacy-Enhanced Mail (Base64 encoded certificate format)
URI	Uniform Resource Identifier
IP	Internet Protocol
VM	Virtual Machine
CDN	Content Delivery Network
DDoS	Distributed Denial of Service
RHEL	Red Hat Enterprise Linux
MBK	Master Backup Key
SO	Security Officer
LAN	Local Area Network
PCIe	Peripheral Component Interconnect Express
DNF	Dandified YUM (package manager for RHEL-based systems)
DN	Distinguished Name
CN	Common Name

Abbreviation	Meaning
EPEL	Extra Packages for Enterprise Linux

Table 1: Abbreviations

1.5 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	Select **Details** and click on **Properties** button
`Monospaced`	Code that is given for explanation or as an example, file paths	`certreq.exe -new request.inf IISCertRequest.csr`
Italic	References and important terms	Operating system listed in *Tested Versions*

Table 2: 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 Cloudflare Keyless SSL

Cloudflare Keyless SSL is a feature of Cloudflare's SSL/TLS service that allows organizations to secure their web traffic using Cloudflare's global edge network while retaining control of their private TLS keys. Instead of storing private keys within Cloudflare's infrastructure, Keyless SSL enables these keys to remain in a customer-managed environment. Cloudflare terminates incoming client connections at its globally distributed edge locations and handles the majority of the TLS handshake process. When a private key operation is required during the handshake, Cloudflare securely communicates with an external key server to complete the operation.

This approach separates TLS processing from private key storage, allowing organizations to leverage Cloudflare's performance and security capabilities such as content delivery optimization, traffic acceleration, and DDoS protection while maintaining full ownership and control over cryptographic keys.

Cloudflare Keyless SSL is designed to integrate with external key management systems through a secure and authenticated connection, ensuring that sensitive key operations are performed without exposing private key material to Cloudflare's infrastructure.

2.2 Utimaco u.trust GP HSM

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

2.3 Joint Value Proposition

The integration of Cloudflare Keyless SSL with Utimaco u.trust GP HSM combines the strengths of Cloudflare's global edge network with the robust security of hardware-based key protection. This approach enables organizations to deliver high-performance, secure web services while ensuring that sensitive cryptographic keys remain protected within a dedicated HSM. By leveraging Cloudflare's edge infrastructure, organizations can benefit from optimized TLS/SSL traffic handling, reduced latency, and built-in protection against Distributed Denial-of-Service (DDoS) attacks. At the same time, the Utimaco u.trust GP HSM ensures that private keys are

securely generated, stored, and used exclusively within the hardware boundary, preventing unauthorized access or key exposure.

This integration provides enhanced compliance with industry standards and regulatory requirements by maintaining strict control over cryptographic key material. It also ensures scalability and operational efficiency, as Cloudflare manages the client-facing traffic while the HSM performs only the necessary cryptographic operations.

Overall, the combined solution delivers a balanced architecture that offers strong security, regulatory compliance, high availability, and improved performance, allowing organizations to meet modern web security and performance demands without compromising on key ownership and control.

3 Integration Requirements and Prerequisites

3.1 Tested Versions

The integrations that have been successfully tested with the u.trust GP HSM with Cloudflare Keyless SSL.

Operating System	Gokeyless	Utimaco Security Server Version	Utimaco HSM
RHEL 9	1.18.0	SecurityServer V6.6.0	u.trust GP HSM CSe-Series/Se-Series

Table 3: List of tested versions



Cloudflare Keyless SSL requires the GoKeyless daemon to enable communication between Cloudflare edge nodes and the Utimaco HSM via PKCS#11. Direct integration without GoKeyless is not supported in this tested setup.

3.2 Software Requirements

Software	Software Requirements
Host VM	Red Hat Enterprise Linux 9 or above
GoKeyless	GoKeyless daemon (v1.18.0 or compatible)
Cloudflare Service	Cloudflare Keyless SSL (active domain + account)
HSM Software	Utimaco SecurityServer (v6.6.0 or compatible)
HSM Interface	Utimaco PKCS#11 library (R3 module configured)

Software	Software Requirements
Network Requirements	Publicly reachable Keyless server (port 2407)
Certificate Requirement	Public CA-signed certificate (e.g., Let's Encrypt, DigiCert)

Table 4: List of software requirements

3.3 Hardware Requirements

Hardware	Hardware Requirements
Utimaco LAN HSM	u.trust GP HSM CSe-Series/Se-Series LAN with firmware SecurityServer 6.6.0 or higher
Utimaco PCI-e HSM	u.trust GP HSM CSe-Series/Se-Series PCI-e with firmware SecurityServer 6.6.0 or higher

Table 5: List of hardware requirements



Set up an account on the Utimaco support portal and request download access at the following URL: https://support.hsm.utimaco.com/ .

3.4 Prerequisites

Before you begin:

- Ensure that the u.trust GP HSM is set up and configured. Ensure the HSM is initialized, reachable, and operational. Refer to the u.trust GP HSM documentation for setup instructions.
- Ensure that the default administrative credentials are replaced. Replace the default admin user with a secure, custom admin user.
- Ensure that the Master Backup Key (MBK) is generated and securely stored. The MBK must be created and stored across the HSM devices as per best practices.

- Ensure that the Supported Operating System is installed. Ensure the host system is running a supported OS as listed in the Tested Versions section (e.g., RHEL 9 or above).
- Ensure that the Utimaco SecurityServer is installed and configured. The SecurityServer version must match the tested version (e.g., v6.6.0), and the HSM should be accessible via the client tools.
- Ensure that the PKCS#11 interface is configured and validated. The Utimaco PKCS#11 library (R3 module) must be installed and correctly configured. Verify access to the HSM using tools such as pkcs11-tool.
- Ensure that the Cloudflare account with Keyless SSL is enabled.
- Ensure the domain is onboarded to Cloudflare and the Keyless SSL feature is available/ enabled in your account.
- Ensure that the network connectivity and firewall rules are configured.
- Ensure that port 2407 is open (default GoKeyless service port). Outbound/inbound connectivity between Cloudflare edge and GoKeyless server must be allowed.
- Ensure there is root/administrative access to the host server. This is required for installing GoKeyless, configuring services, and managing certificates.

4 Installation and Configuration of Utimaco SecurityServer Software

4.1 Downloading and Installing Utimaco Software

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

1. Copy the downloaded software to the appropriate location on the Cloudflare Keyless SSL Server.
2. Create the required directories under `/opt/utimaco`.

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

3. Copy the PKCS#11 library (`libcs_pkcs11_R3.so`) to the library directory.

```
sudo cp ~/path_to_application_folder/lib/libcs_pkcs11_R3.so /opt/utimaco/lib/
```

4. Copy the required Utimaco utilities (`csadm` and `p11tool2`) to the binary directory.

```
cd ~/path_to_application_folder
sudo cp csadm p11tool2 /opt/utimaco/bin/
```

5. Make the binaries executable.

```
sudo chmod +x /opt/utimaco/bin/csadm /opt/utimaco/bin/p11tool2
```

4.2 u.trust GP HSM PKCS#11 Configuration

1. Create a directory to store the PKCS#11 configuration file.

```
sudo mkdir -p /opt/utimaco/PKCS11_R3
```

2. Navigate to the PKCS#11 sample configuration directory in the Utimaco software package and copy the configuration file.

```
cd <install_directory>/Software/Linux/x86-64/Crypto_APIS/PKCS11_R3/sample
sudo cp cs_pkcs11_R3.cfg /opt/utimaco/PKCS11_R3/
```

3. Move to the configuration directory.

```
cd /opt/utimaco/PKCS11_R3
```

4. Update the PKCS#11 configuration fileEdit the configuration file.

```
sudo vi /opt/utimaco/PKCS11_R3/cs_pkcs11_R3.cfg
```

Update the configuration as shown below.

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

5. Set proper permissions (required for GoKeyless).

```
sudo chown -R gokeyless:gokeyless /opt/utimaco/PKCS11_R3
```

6. Run the following command to validate HSM connectivity and PKCS#11 configuration.

```
./p11tool2 ListSlots
```

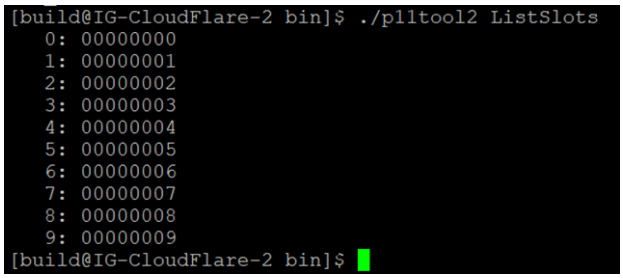


Figure 1 : Listing available HSM slots with p11tool2

For more information regarding the commands and command parameters, please check the u.trust GP HSM documentation. The device may be a u.trust GP HSM (PCIe or LAN) device.
The device line will follow one of these patterns, based on the HSM form-factor:

Device = 288@<HSM IP address> Hardware (LAN) HSM
OR
Device = /dev/cs2.0 Hardware (PCIe) HSM

To make your testing easier, you can enable the PKCS#11 log file.
That can be enabled by editing the Logging Loglevel. Set the LogPath and Logging Loglevel to 1. For testing you may want to increase it to 4.
The added LogPath points to a writable directory, not to a file.
If you encounter problems, check the log file named `cs_pkcs11_R3.log` in the LogPath defined directory. When you are done testing, you should change Logging to 1 or 2. This will limit the logging to only critical and important messages.

4.3 u.trust GP HSM Simulator Setup and Initialization

This section describes how to set up and start the u.trust GP HSM Simulator on the Keyless server for testing purposes.

1. Copy the simulator package (sim5_linux) to the designated directory on the server.

```
sudo mkdir -p /opt/utimaco/Simulator
sudo cp -r <path_to_sim5_linux> /opt/utimaco/Simulator/
```

2. Assign execution permissions to the simulator binaries.

```
cd /opt/utimaco/Simulator/sim5_linux
sudo chmod +x *
```

3. Install the required 32-bit libraries (if not already installed).

```
sudo dnf update -y
sudo dnf install glibc.i686 libstdc++.i686 zlib.i686 -y
```

4. Verify required system library.

```
ls /lib/ld-linux.so.2
```

The file path should be displayed. If not, install missing dependencies.

5. Navigate to the simulator directory and start the simulator.

```
cd /opt/utimaco/Simulator/sim5_linux
./bl_sim5 -h -o -d ../devices
```

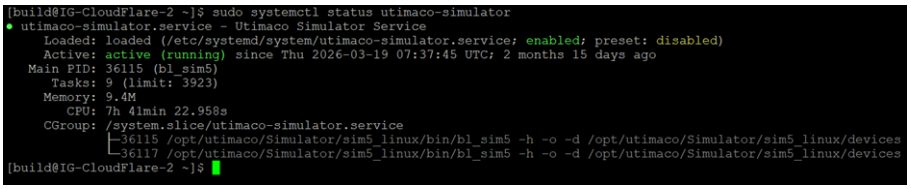


Figure 2 : Utimaco HSM simulator successfully started

The simulator setup described in this guide is optional and intended for lab and testing purposes only.

For production deployments, it is recommended to use a physical u.trust GP HSM, as it provides the required security, performance, and compliance characteristics expected in a production environment.

4.4 Create SO User and Initialize a Slot

This section describes how to create Security Officer (SO) and User accounts, initialize a slot, and update default credentials on the u.trust GP HSM Simulator.

Prerequisites:

- the HSM Simulator is running and accessible (e.g., `127.0.0.1`).
- the `csadm` utility is available in the current directory or PATH.
- the admin key file (e.g., `ADMIN_SIM.key`) is available.

Steps:

1. Create a user assigned to the slot.

```
./csadm dev=3001@127.0.0.1 LogonSign=ADMIN,./ADMIN_SIM.key
AddUser=USR_0000,00000022{CXI_GROUP=SLOT_0000},hmacpwd,87654321
```

2. Create the SO user for slot management.

```
./csadm dev=3001@127.0.0.1 LogonSign=ADMIN,./ADMIN_SIM.key
AddUser=SO_0000,00000200{CXI_GROUP=SLOT_0000},hmacpwd,87654321
```

3. Change the default password for the SO user.

```
./csadm dev=3001@127.0.0.1 Logonpass=SO_0000,87654321 ChangeUser=SO_0000,Cloud123
```

4. Change the default password for the user.

```
./csadm dev=3001@127.0.0.1 Logonpass=USR_0000,87654321
ChangeUser=USR_0000,Cloud123
```

5. List all users to confirm successful creation.

```
./csadm dev=3001@127.0.0.1 LogonSign=ADMIN,./ADMIN_SIM.key ListUsers
```

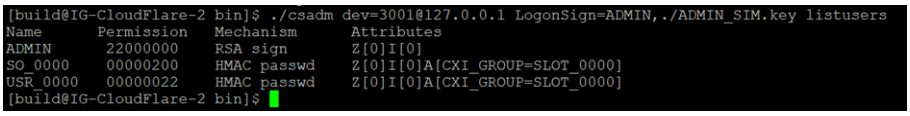


Figure 3 : Utimaco HSM user configuration details

- Replace `127.0.0.1` with the actual HSM IP address if using a remote device.
- Replace passwords (`87654321` , `Cloud123`) with secure values as per policy.
- Ensure the slot name (`SLOT_0000`) matches your configuration.
- The group assignment `{CXI_GROUP=SLOT_0000}` links the user to the corresponding slot.

The SO is responsible for slot initialization and user management, while the User account is used for cryptographic operations via PKCS#11 (e.g., GoKeyless integration).

4.5 Installation and Configuration of Cloudflare Keyless SSL

4.5.1 Installation of GoKeyless

1. Update the System Packages. First, ensure that the system package repository is up to date.

```
sudo dnf update -y
```

This command updates all installed packages to their latest versions to ensure compatibility and avoid dependency-related issues during installation.

2. Install the necessary tools required for downloading and managing packages.

```
sudo dnf install -y wget curl tar
```

`wget` and `curl` are used for downloading files from external sources. `tar` is used for extracting compressed archives if required during setup.

3. Install the DNF plugins package and add the Cloudflare GoKeyless repository.

```
sudo dnf install -y dnf-plugins-core
sudo dnf config-manager --add-repo https://pkg.cloudflare.com/gokeyless.repo
```

`dnf-plugins-core` enables repository management features. The Cloudflare repository is added to allow installation of the GoKeyless package directly using the package manager.

4. Install the GoKeyless package from the configured repository.

```
sudo dnf install -y gokeyless
```

This installs the Cloudflare GoKeyless daemon, which acts as a bridge between Cloudflare's edge servers and the private key stored in the HSM.

```
[root@LinuxVM-cloudflare-New build]# sudo dnf install -y gokeyless
gokeyless-stable                                                                                    1.2 kB/s | 1.1 kB     00:00
Dependencies resolved.
================================================================================================================================
 Package                    Architecture            Version                     Repository                                 Size
================================================================================================================================
Installing:
 gokeyless                  x86_64                  1.18.0+bookworm-1           gokeyless-stable                           15 M
Installing dependencies:
 libtool-ltdl               x86_64                  2.4.6-45.el9                rhel-9-for-x86_64-appstream-eus-rhui-rpms   39 k

Transaction Summary
================================================================================================================================
Install  2 Packages

Total download size: 15 M
Installed size: 33 M
Downloading Packages:
(1/2): libtool-ltdl-2.4.6-45.el9.x86_64.rpm                                          45 kB/s |  39 kB     00:00
(2/2): gokeyless-1.18.0+bookworm-1.x86_64.rpm                                        15 MB/s |  15 MB     00:01
--------------------------------------------------------------------------------------------------------------------------------
Total                                                                                15 MB/s |  15 MB     00:01
gokeyless-stable                                                                    9.9 kB/s | 3.1 kB     00:00
Importing GPG key 0x8D4E5E73:
 Userid     : "CloudFlare Software Packaging 2025 <help@cloudflare.com>"
 Fingerprint: CC94 B39C 77AE 7342 A68B 8962 8A68 2D30 8D4E 5E73
 From       : https://pkg.cloudflare.com/cloudflare-ascii-pubkey.gpg
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                    1/1
  Installing       : libtool-ltdl-2.4.6-45.el9.x86_64                                                                   1/2
  Running scriptlet: gokeyless-1.18.0+bookworm-1.x86_64                                                                 2/2
  Installing       : gokeyless-1.18.0+bookworm-1.x86_64                                                                 2/2
  Running scriptlet: gokeyless-1.18.0+bookworm-1.x86_64                                                                 2/2
/var/tmp/rpm-tmp.Z1ObUW: line 1: /sbin/chkconfig: No such file or directory

  Verifying        : gokeyless-1.18.0+bookworm-1.x86_64                                                                 1/2
  Verifying        : libtool-ltdl-2.4.6-45.el9.x86_64                                                                   2/2
Installed products updated.
```

Figure 4 : GoKeyless setup and package installation

5. Configure file permissions.

```
sudo chmod 644 /etc/keyless/gokeyless.yaml
```

6. Add the required user (e.g., `build`) to the `keyless` group.

```
sudo usermod -aG keyless build
```

7. Check the installed GoKeyless version.

```
gokeyless --version
```

```
[root@LinuxVM-cloudflare-New build]# gokeyless --version
gokeyless version 1.18.0+bookworm
github.com/cloudflare/gokeyless/cmd/gokeyless built with go1.26.2
```

Figure 5 : GoKeyless version details

4.5.2 Cloudflare Dashboard Configuration

1. Log in to the Cloudflare Dashboard.
   Navigate to the Cloudflare portal and log in using your credentials.

2. Select the Domain.
   After logging in, select the required domain from the list of configured domains. This ensures that all subsequent configurations are applied to the correct domain associated with the Keyless SSL setup.

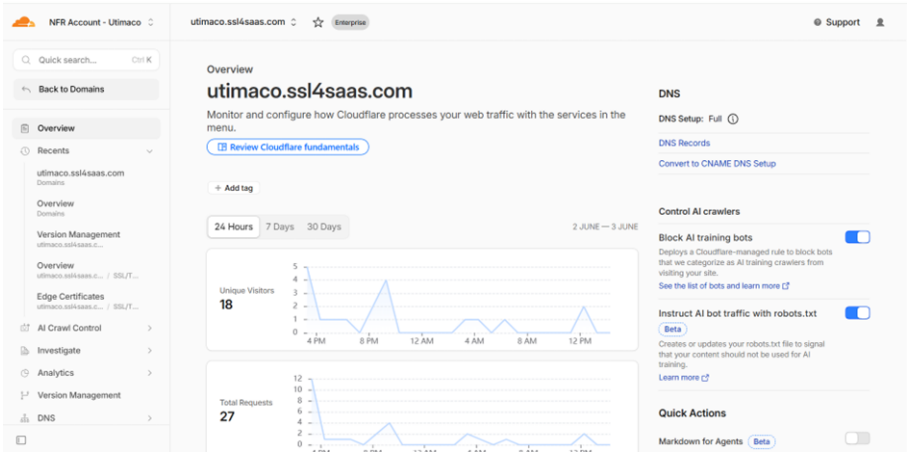


Figure 6 : Cloudflare domain overview dashboard

3. Navigate to the DNS Configuration.
   Go to the DNS section of the selected domain. The DNS section is used to manage domain

records, which are required to route traffic correctly between Cloudflare, the key server, and the origin server.

4. **Create DNS Records.**
   Create the required DNS records for both the key server and user-facing application.

Key Server DNS Record

```
Hostname: kms.utimaco.ssl4saas.com
Type: A
Proxy Status: DNS only (Grey Cloud)
```

The key server hostname must be configured as **DNS-only** to allow Cloudflare to directly communicate with the GoKeyless server over port 2407 without proxy interference.

User Access DNS Record

```
Hostname: utimaco.ssl4saas.com
Type: A
Proxy Status: Proxied (Orange Cloud)
```

The user-facing hostname must be **proxied through Cloudflare** to enable SSL termination at the Cloudflare edge and to utilize the Keyless SSL configuration.

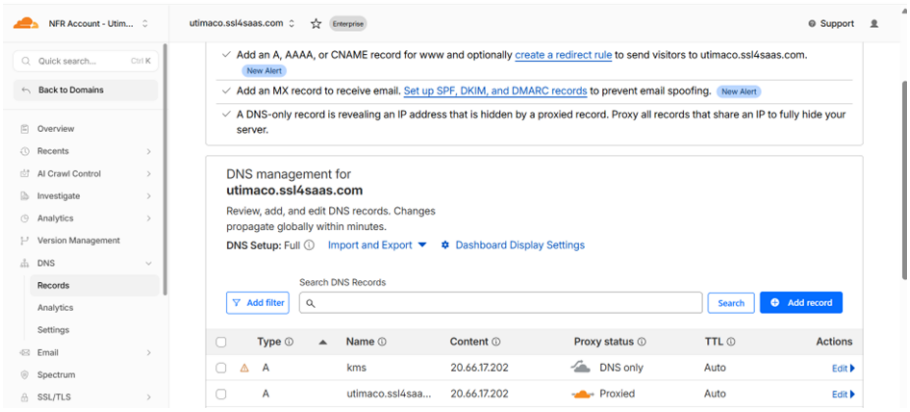


Figure 7 : Cloudflare DNS records configuration

5 Integration Steps

5.1 Generating Key on Utimaco HSM

1. Generate the key pair using the below `p11tool2` command.

```
./p11tool2  Slot=0 LoginUser=Cloud123
PubKeyAttr=CKA_LABEL="rsa_public_key",CKA_MODULUS_BITS=2048,CKA_ID="01"
PrvKeyAttr=CKA_LABEL="rsa_private_key",CKA_EXTRACTABLE=CK_FALSE,CKA_ID="01"
GenerateKeyPair=RSA
```

2. Verify that the keys are generated.

```
./p11tool2 Slot=0 LoginUser=Cloud123 ListObjects
```

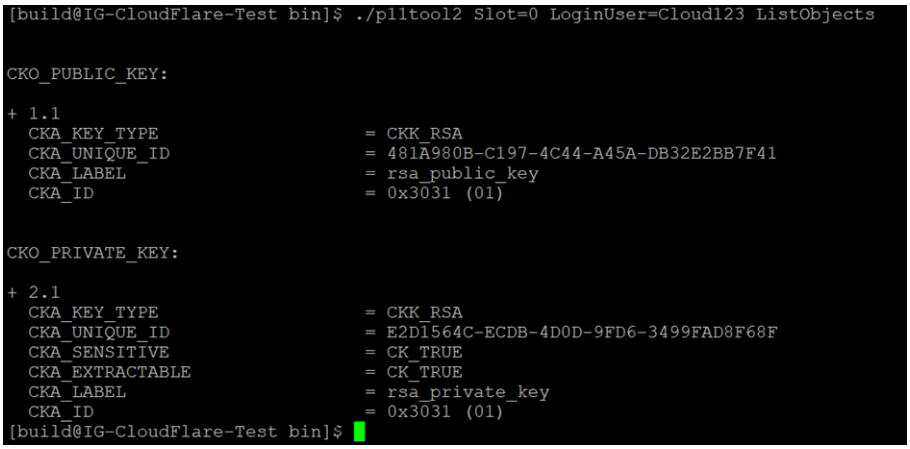


Figure 8 : Viewing keys in HSM slot using p11tool

5.2 CSR Generation Using HSM-Based Private Key

1. Install `openssl`.

```
sudo dnf install openssl -y
openssl version
```

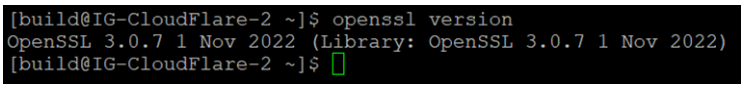


Figure 9 : openssl version

2. Generate the CSR.

```
./p11tool2 Slot=0 LoginUser=Cloud123
PubKeyAttr=CKA_LABEL="rsa_public_key",CKA_MODULUS_BITS=2048,CKA_ID="01"
PrvKeyAttr=CKA_LABEL="rsa_private_key",CKA_EXTRACTABLE=CK_FALSE,CKA_ID="01"
Mech=CKM_SHA256_RSA_PKCS   DN='O=Integration,CN=utimaco.ssl4saas.com'
ExportP10=/tmp/testhsm_v1.csr
```

This command generates a CSR using an RSA key pair stored in the HSM by specifying key attributes, authentication details, and the required certificate subject information.

DN='O=Integration,CN=utimaco.ssl4saas.com'

Defines the subject name for the CSR:

- `O` → Organization name (Integration).
- `CN` → Common Name (domain name used in SSL certificate).

5.3 Signing CSR with Public Certificate Authority

This section outlines the general process of submitting the generated Certificate Signing Request (CSR) to a trusted Public Certificate Authority (CA) to obtain a signed SSL/TLS certificate.

The signed certificate is deployed at the Cloudflare edge to enable secure TLS communication as part of the Keyless SSL setup.

1. **Select a Public Certificate Authority.**

- Choose a trusted CA based on organizational policies and production requirements.
- Ensure the CA supports the required domain validation methods (e.g., DNS, HTTP, or email validation).

2. **Submit the CSR.**

- Provide the generated CSR file to the selected CA through their portal or API.
- Ensure that all required domain names (including SAN entries, if applicable) are correctly included in the CSR.

3. **Complete Domain Validation.**

- Perform domain ownership verification as required by the CA.
- This may involve adding DNS records, hosting validation files, or responding to validation emails.

4. **Certificate Issuance.**

- After successful validation, the CA issues the signed certificate.
- The certificate bundle may include:
    - Server certificate.
    - Intermediate CA certificate(s).
    - Root CA certificate (optional, depending on the CA).

5. **Download the Certificate Chain.**

- Retrieve the signed certificate along with the full certificate chain from the CA.
- Ensure the correct format is used (e.g., PEM).

6. **Prepare for Deployment.**

- Verify that the certificate matches the private key stored in the HSM.
- Confirm all required domains are covered and the certificate validity period meets operational requirements.



The choice of Certificate Authority and certificate lifecycle management (renewal, revocation, etc.) should align with organizational security policies.

5.4 Upload the Keyless SSL Certificate

This step sets up the Keyless SSL configuration in Cloudflare and uploads the signed certificate, allowing secure communication with the Keyless server for private key operations.

1. Log in to the **Cloudflare Dashboard** and select the target domain (zone).
2. Navigate to **SSL/TLS → Edge Certificates**.
3. Click on **Upload Keyless SSL Certificate** under **Manage Edge Certificates**.
4. In the configuration form, provide the following details.

- **Key Server Label.**
  Enter a descriptive label to identify the Keyless server (e.g., `keyless`).
- **Key Server Hostname.**
  Specify the fully qualified domain name (FQDN) of the Keyless server.
  Example: `kms.utimaco.ssl4saas.com`.
- **Key Server Port.**
  Enter the listening port (2407) configured for the GoKeyless service.
- **SSL Certificate.**
  Paste the **signed domain certificate** (Refer to section 5.3).

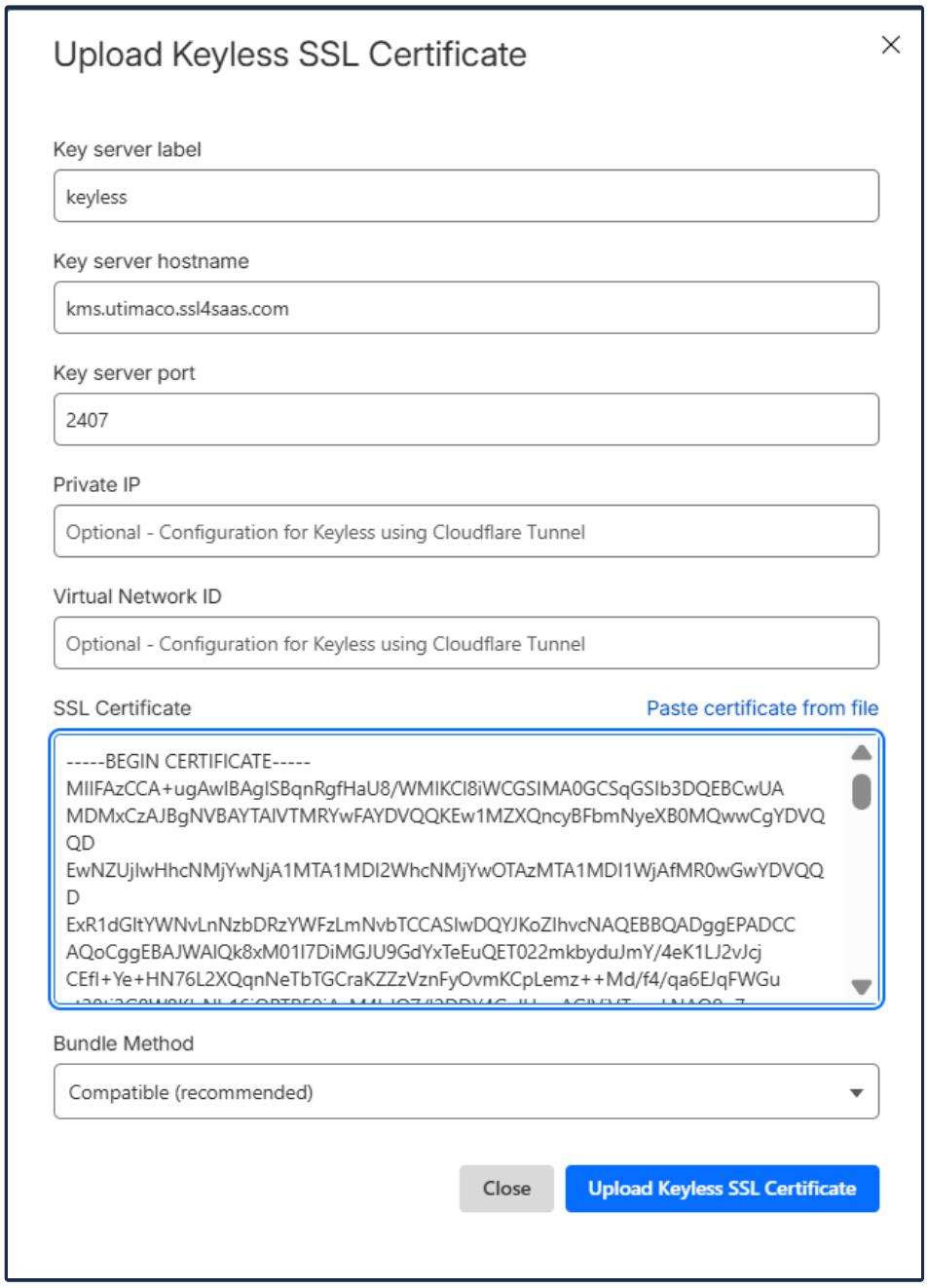


Figure 10 : Uploading keyless SSL certificate in Cloudflare

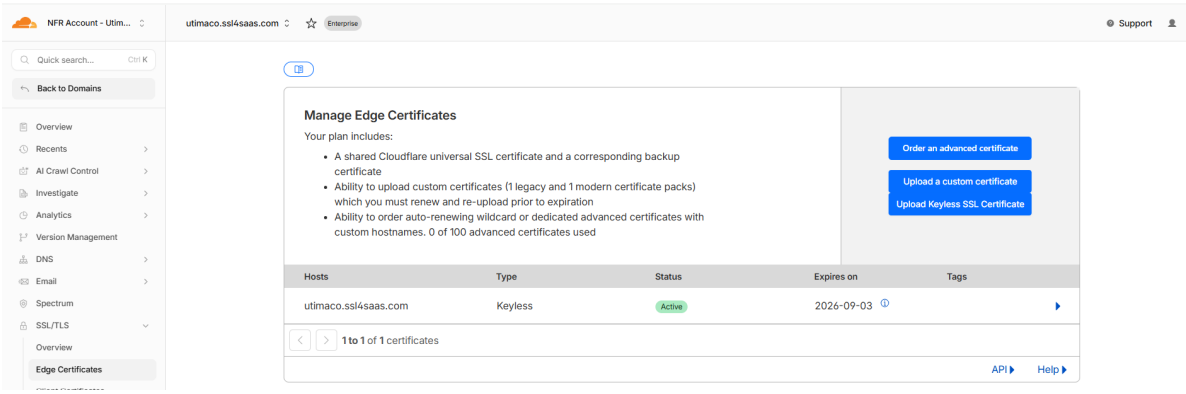


Figure 11 : Viewing keyless SSL certificate status in Cloudflare

In this validation setup, Cloudflare SSL/TLS mode was set to **Flexible**, meaning HTTPS was enforced between the client and Cloudflare, while communication between Cloudflare and the origin server remained **HTTP**.

Customers using **HTTPS** at the origin should configure Cloudflare SSL mode as **Full** or **Full (Strict)** to ensure end-to-end encryption. The selection depends on whether a valid CA-signed certificate is installed on the origin server.

Misconfiguration of SSL mode can result in handshake failures, redirect loops, or insecure communication paths.

5.5 GoKeyless Configuration

This section describes the configuration of the GoKeyless service by updating the `gokeyless.yaml` file with environment-specific parameters, including Cloudflare account details, HSM configuration, and authentication certificate paths.

1. Update the GoKeyless Configuration File. Open the GoKeyless configuration file using the following command.

```
sudo nano /etc/keyless/gokeyless.yaml
```

The output is as follows:

```
# Set the log level (0 = DEBUG, 5 = FATAL).
loglevel: 1

# Hostname must match the key server hostname that was configured in the
```

```
# Cloudflare dashboard during custom certificate upload.
hostname:

# Zone ID can be found on the Cloudflare dashboard 'Overview' tab.
zone_id:

# Preferred credential used to request the server authentication certificate:
# a Cloudflare API token scoped Zone > SSL and Certificates > Edit. See
# https://developers.cloudflare.com/fundamentals/api/get-started/create-token/
api_token:

# Deprecated: the legacy Origin CA Service Key (found on the Cloudflare
# dashboard under the 'My Profile' section). Prefer api_token above. Only one
# of api_token or origin_ca_api_key needs to be set.
origin_ca_api_key:

# Configure one or more private key directories.
private_key_stores:- uri: "pkcs11:slot-
id=0;object=rsa_private_key;id=%30%31;type=private?module
path=/opt/utimaco/lib/libcs_pkcs11_R3.so&pin-value=Cloud123"
#- dir: /etc/keyless/keys

# Optionally customize the location of the certificates used for mutual
# authentication with Cloudflare keyless clients.
auth_cert: /etc/keyless/server.pem
auth_key: /etc/keyless/server-key.pem
auth_csr: /etc/keyless/server.csr
cloudflare_ca_cert: /etc/keyless/keyless_cacert.pem

# Optionally customize the listen ports.
port: 2407
metrics_port: 2406

# Optionally write the PID to a file (note that sysv-based systems will
# ignore this value and always use /var/run/gokeyless.pid).
pid_file:
```

2. Save the changes and exit.



Note: When using GoKeyless v1.18 or later, configure a Cloudflare API Token instead of an Origin CA API Key, as v1.18 adds support for API Token-based authentication

6 Verification and Testing

6.1 Service Startup and Integration Validation

6.1.1 Nginx Installation and Configuration

This section describes the installation and configuration of the Nginx web server, which is used to handle HTTP/HTTPS requests and assist in validating the Keyless SSL setup.

1. Install the Nginx package using the following command:

```
sudo dnf install -y nginx
```

2. Enable the Nginx service to start automatically on system boot, and start the service.

```
sudo systemctl enable nginx
sudo systemctl start nginx
sudo systemctl status nginx
```

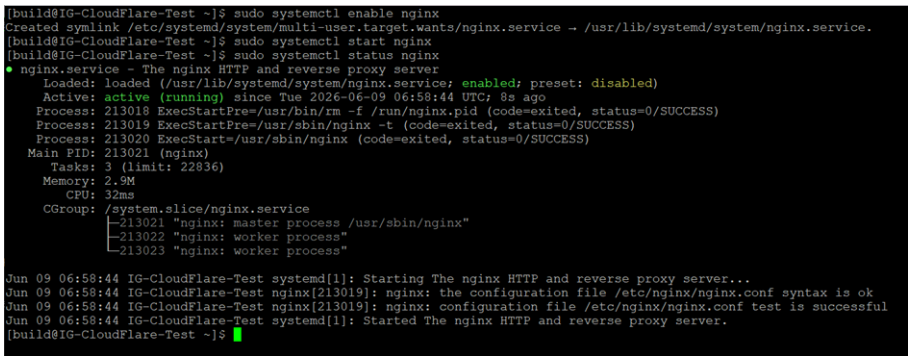


Figure 12 : Starting and verifying nginx service

3. Open the required firewall ports to allow HTTP and HTTPS traffic.

```
sudo firewall-cmd --add-service=http --permanent
sudo firewall-cmd --add-service=https --per
```

- Ports **80 (HTTP)** and **443 (HTTPS)** must be accessible to allow incoming client connections.
- This ensures that the web server can receive traffic for testing and validation of the SSL setup.

4. Verify that Nginx is successfully listening on the required ports.

```
ss -tulnp | grep -E '80|443'
```

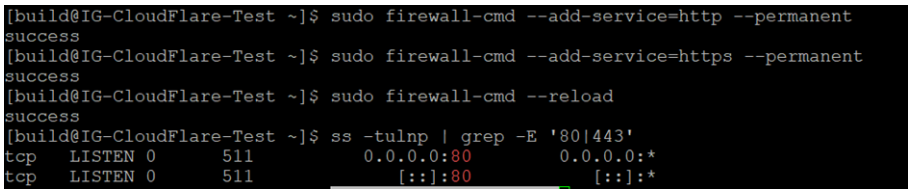


Figure 13 : Firewall configuration and verification of HTTP/HTTPS ports

6.1.2 GoKeyless Service Setup and Initialization

This section describes the creation of a `systemd` service for the GoKeyless server and the steps required to start, enable, and verify the service. Configuring GoKeyless as a system service ensures it runs consistently and starts automatically during system boot.

1. Create `systemd` service for GoKeyless.

```
sudo nano /usr/lib/systemd/system/gokeyless.service
```

Add the following configuration:

```
[Unit]
Description=gokeyless daemon
After=network.target

[Service]
Type=simple
User=gokeyless
Group=gokeyless
WorkingDirectory=/etc/keyless
ExecStart=/usr/bin/gokeyless
```

```
[Install]
WantedBy=multi-user.target
```

2. Reload `systemd` configuration.

```
sudo systemctl daemon-reexec
sudo systemctl daemon-reload
```

3. Enable and start the GoKeyless service.

```
sudo systemctl enable gokeyless
sudo systemctl start gokeyless
```

4. Verify service status.

```
sudo systemctl status gokeyless
```

```
● gokeyless.service - gokeyless daemon
     Loaded: loaded (/usr/lib/systemd/system/gokeyless.service; enabled; preset: disabled)
    Drop-In: /etc/systemd/system/gokeyless.service.d
             └─override.conf
     Active: active (running) since Thu 2026-10-01 09:59:45 UTC; 2 weeks 3 days ago
   Main PID: 830182 (gokeyless)
      Tasks: 8 (limit: 4212)
     Memory: 43.4M
        CPU: 2min 42.224s
     CGroup: /system.slice/gokeyless.service
             └─830182 /usr/bin/gokeyless
```

Figure 14 : Monitoring GoKeyless daemon status

6.2 Logs and Validation Steps

1. Run the following command to verify the GoKeyless log verification.

```
journalctl -u gokeyless -f
```

Logs:

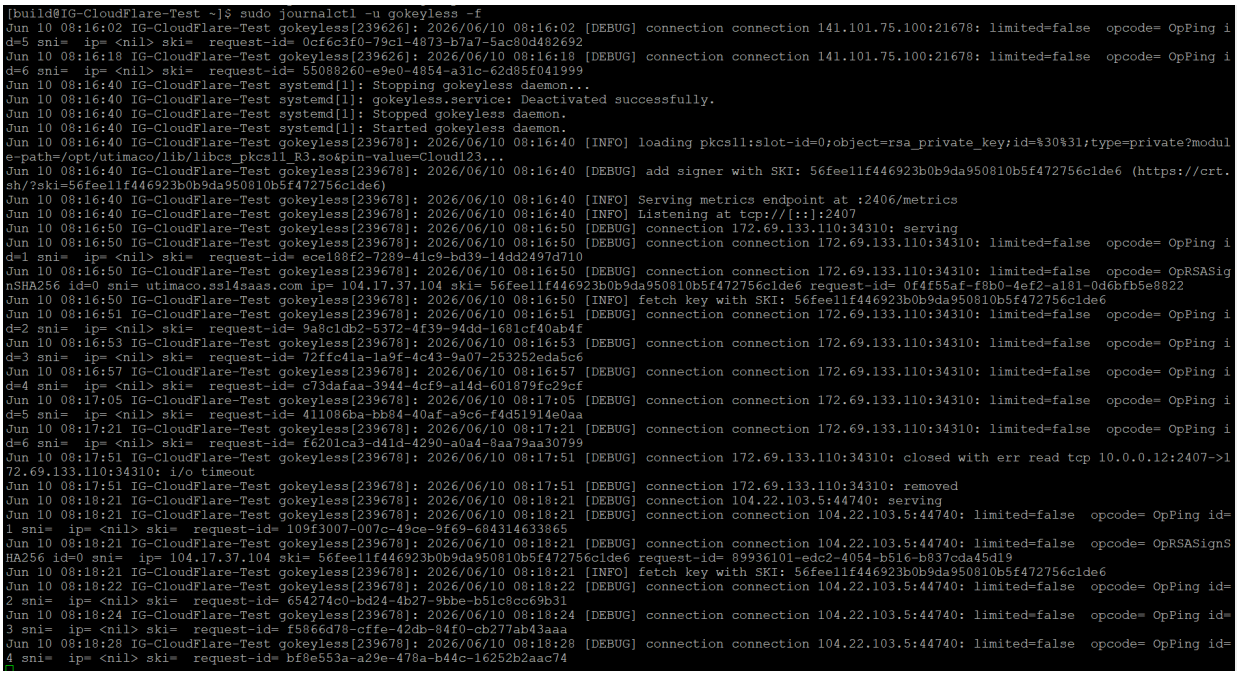


Figure 15 : GoKeyless logs

```
curl -v https://utimaco.ssl4saas.com
```

```
[build@IG-CloudFlare-Test ~]$ curl -v https://utimaco.ssl4saas.com
*   Trying 104.17.37.104:443...
* Connected to utimaco.ssl4saas.com (104.17.37.104) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
*  CAfile: /etc/pki/tls/certs/ca-bundle.crt
* TLSv1.0 (OUT), TLS header, Certificate Status (22):
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS header, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS header, Certificate Status (22):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS header, Certificate Status (22):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS header, Certificate Status (22):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS header, Certificate Status (22):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS header, Finished (20):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS header, Certificate Status (22):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS header, Finished (20):
* TLSv1.2 (IN), TLS header, Certificate Status (22):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-CHACHA20-POLY1305
* ALPN, server accepted to use h2
* Server certificate:
*  subject: CN=utimaco.ssl4saas.com
*  start date: Jun  5 10:50:26 2026 GMT
*  expire date: Sep  3 10:50:25 2026 GMT
*  subjectAltName: host "utimaco.ssl4saas.com" matched cert's "utimaco.ssl4saas.com"
*  issuer: C=US; O=Let's Encrypt; CN=YR2
*  SSL certificate verify ok.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* TLSv1.2 (OUT), TLS header, Unknown (23):
* TLSv1.2 (OUT), TLS header, Unknown (23):
* TLSv1.2 (OUT), TLS header, Unknown (23):
* Using Stream ID: 1 (easy handle 0x55e05f742d20)
* TLSv1.2 (OUT), TLS header, Unknown (23):
> GET / HTTP/2
> Host: utimaco.ssl4saas.com
> user-agent: curl/7.76.1
> accept: */*
>
* TLSv1.2 (IN), TLS header, Unknown (23):
* TLSv1.2 (OUT), TLS header, Unknown (23):
```

Figure 16 : TLS handshake and server response validation using curl command

Note: GoKeyless v1.18 accepts only TLS 1.2 connections on the key server interface (port 2407). TLS 1.3 connection attempts are expected to fail during the handshake. Successful Cloudflare-to-GoKeyless communication uses TLS 1.2 with mutual TLS authentication.

7 Troubleshooting

7.1 Common issues and how to resolve them

Error	Diagnosis
LoginUser= failed: 05.12.2021 23:45:45 src/p11adm_R2.c[429] p11_login: C_Login [type=1] returned Error 0x00000102 (CKR_USER_PIN_NOT_INITIALIZED)	PKCS#11 slot is not initialized.
The CryptoServer PKCS#11 Library R3 is not initialized. Error CKR_CRYPTOKI_NOT_INITIALIZED occurred.	PKCS#11 slot is not initialized.
TLS handshake failed: tls: client offered only unsupported versions: [304]	Enable DEBUG logging. Disable TLS 1.3 in cloudflare.

Table 6: List of errors and their diagnoses

7.2 Log locations and interpretation

The log file locations may vary based on the configured file paths; the Utimaco HSM log location is defined in the `cs_pkcs11_R3.cfg` configuration file, while the GoKeyless log output is configured within the `gokeyless.yaml` file and can also be accessed via system logs (systemd journal).

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
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.

9 Appendices

9.1 Command Summary (CLI commands used)

Command	Purpose
`sudo nano /etc/keyless/gokeyless.yaml`	Edit the GoKeyless configuration file to provide Cloudflare, HSM, and authentication parameters.
`sudo nano /usr/lib/systemd/system/gokeyless.service`	Create or modify the systemd service file for GoKeyless.
`sudo systemctl daemon-reexec`	Re-execute systemd manager to apply configuration changes.
`sudo systemctl daemon-reload`	Reload systemd configuration after creating or modifying service files.
`sudo systemctl enable gokeyless`	Enable GoKeyless service to start automatically on system boot.
`sudo systemctl start gokeyless`	Start the GoKeyless service.
`sudo systemctl status gokeyless`	Check the status and health of the GoKeyless service.
`journalctl -u gokeyless -f`	View real-time logs for GoKeyless service (used for validation and troubleshooting).
`sudo dnf install -y nginx`	Install Nginx web server package.

Command	Purpose
`sudo systemctl enable nginx`	Enable Nginx service to start automatically on boot.
`sudo systemctl start nginx`	Start the Nginx service.
`sudo systemctl status nginx`	Verify that Nginx is running successfully.
`sudo firewall-cmd --add-service=http --permanent`	Allow HTTP (port 80) traffic through the firewall.
`sudo firewall-cmd --add-service=https --permanent`	Allow HTTPS (port 443) traffic through the firewall.
`sudo firewall-cmd --reload`	Reload firewall rules to apply changes.
`` `ss -tulnp ``	grep -E '80
`` `ss -tulnp ``	grep 2407`
`curl -v utimaco.ssl4saas.com`	Validate HTTPS connectivity and inspect TLS handshake details.
`openssl s_client -connect <domain>:443`	Perform detailed TLS handshake validation and certificate inspection.
`pkcs11-tool --module <module_path> -O`	List objects (keys/certificates) available in the HSM.

Command	Purpose
`journalctl -u gokeyless --since "10 min ago"`	View recent GoKeyless logs for troubleshooting.

Table 7: List of commands