# Microsoft

# JSign

7.4

# Integration Guide

# u.trust GP HSM

6.5.0

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 | 2026-07-15 |
| Status | PUBLISHED |
| Document No. | IG-2026-0063 |

# Table of Contents

# 1 About This Guide

This guide describes how to enable u.trust GP HSM integration with JSign.

## 1.1 Target Audience

This guide is intended for JSign and Utimaco GP HSM administrators.

## 1.2 Purpose of the Integration

The purpose of integrating JSign with Utimaco u.trust HSM is to enable secure Authenticode code signing by allowing JSign to use cryptographic keys stored and protected within the Utimaco HSM through the PKCS#11 interface. This integration ensures that private signing keys remain inside the HSM and are never exposed to the operating system or build environment, while providing a scalable, automated, and compliant solution for signing Windows executables.

## 1.3 Abbreviations

| Abbreviation | Meaning |
|---|---|
| HSM | Hardware Security Module |
| PKI | Public Key Infrastructure |
| TDE | Transparent Data Encryption |
| PKCS | Public Key Cryptography Standards |
| PKCS#11 | PKCS Part 11: The Cryptographic Token Interface Standard |
| SO | Security Officer |
| USR | Crypto User |

| Abbreviation | Meaning |
|---|---|
| DB | Database |
| JRE | Java Runtime Environment |
| JDK | Java Development Kit |
| JAR | Java Archive |
| MBK | Master Backup Key |
| P11CAT | PKCS#11 graphical interface tool |
| CXI | Cryptographic eXtended Interface |
| FIPS | Federal Information Processing Standards |
| CSR | Certificate Signing Request |
| CA | Certificate Authority |
| DLL | Dynamic Link Library |
| CAB | Cabinet File |
| SDK | Software Development Kit |
| PIN | Personal Identification Number |
| RSA | Rivest–Shamir–Adleman |

| Abbreviation | Meaning |
|---|---|
| CN | Common Name |
| DN | Distinguished Name |
| TSA | Time Stamp Authority |
| CLI | Command Line Interface |
| AIA | Authority Information Access |
| CRL | Certificate Revocation List |
| DER | Distinguished Encoding Rules |
| PEM | Privacy-Enhanced Mail |

Table 1: 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 | Press **OK** |
| `Monospaced` | Code that is given for explanation or as an example, file paths | `chsm-create` |

| Convention | Use | Example |
|---|---|---|
| *Italic* | References and important terms | See *Sample Chapter* in the *CryptoServer - Sample Manual* |

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 Overview JSign

JSign is a Java-based implementation of Microsoft Authenticode used to digitally sign Windows executables, installers, and scripts. It is a cross-platform alternative to Microsoft's signtool.exe, allowing code signing from Windows, Linux, and macOS.

## 2.2 Overview of Utimaco u.trust GP HSM Se-Series

The Utimaco u.trust GP HSM Se-Series 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. The u.trust GP HSM can be used as a universal, independent security component for heterogeneous computer systems.

## 2.3 Joint Value Proposition

JSign integrated with Utimaco u.trust HSM provides a secure, HSM-backed code-signing solution that enables organizations to digitally sign Windows executables, installers, and scripts while ensuring that private signing keys remain protected within the HSM and never leave the secure boundary. By combining JSign's cross-platform Authenticode signing capabilities with Utimaco's PKCS#11-based key management, organizations can implement automated, scalable, and compliant code-signing processes across DevOps and CI/CD environments, reducing the risk of key compromise while maintaining software integrity, authenticity, and trust.

# 3 Integration Requirements and Prerequisites

## 3.1 Tested Versions

| Operating System | Windows SDK | Java | JSign | Utimaco SecurityServer Version | Utimaco HSM |
|---|---|---|---|---|---|
| Windows Server 2022 Windows 11 | 10.0.28000.0 | JDK 11.1 | 7.4 | 6.5.0 | u.trust GP HSM Se-Series |

Table 3: Tested versions

## 3.2 Supported Platforms

The integration was tested and validated on the platforms and software versions listed below. These platforms represent the environment used during integration testing of JSign with Utimaco u.trust HSM using the PKCS#11 interface.

## 3.3 Hardware and Software Requirements

| Hardware | Hardware Requirements |
|---|---|
| Utimaco LAN HSM | u.trust GP HSM Se-Series LAN with firmware SecurityServer 6.5.0 or higher |
| Utimaco PCI-e HSM | u.trust GP HSM Se-Series PCI-e with firmware SecurityServer 6.5.0 or higher |

Table 4: List of hardware requirements

| Software | Software Requirements |
|---|---|
| HSM Interface | SecurityServer PKCS#11 Provider |

| Software | Software Requirements |
|---|---|
| HSM Utility | SecurityServer PKCS#11 Tool (p11tool2) |
| JSign | 7.4 |
| Java | JDK 11.1 |
| Windows SDK | 10.0.28000.0 |

Table 5: List of software requirements

## 3.4 Prerequisites

Before proceeding, ensure you have:

- Installed and configured the operating system listed in Tested Versions.
- Installed and configured the SecurityServer version listed in Tested Versions.
- Replaced the GP HSM Default Admin with a new admin user.
- Created and stored the MBK onto each HSM. Refer to the u.trust HSM documentation to set up the MBK.
- Set up and configured the GP HSM. Refer to the u.trust HSM documentation to set up the HSM.
- Installed and configured Microsoft SDK as listed in Tested Versions.

# 4 Installing and Configuring Utimaco SecurityServer Software

The following section outlines the procedures required to configure the Utimaco SecurityServer Software.

If you have not already done so, create and request an Utimaco Support Portal Account at https://support.hsm.utimaco.com/support. This will allow you to download the software components needed for this installation.

On Linux:

1. Copy the downloaded software to the appropriate location on the Partner Product Server.
2. Create a `utimaco` folder under the `/opt` directory and create two directories `/opt/utimaco/bin` and `/opt/utimaco/lib`.

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

3. Copy the pkcs11 library file `libcs_pkcs11_R3.so` from the Utimaco CryptoServer software to the `/opt/utimaco/lib` directory and make the file executable.

```
cp ~/path_to_application_folder/lib/libcs_pkcs11_R3.so /opt/utimaco/lib
chmod +x /opt/utimaco/lib/libcs_pkcs11_R3.so
```

4. Copy the `csadm` and `p11tool2` files from the Utimaco CryptoServer software to the `/opt/utimaco/bin` directory and make both files executable.

```
# cd ~/path_to_application_folder
# cp csadm p11tool2 /opt/utimaco/bin
# chmod +x /opt/utimaco/bin/csadm /opt/utimaco/bin/p11tool2
```

5. Create the directory `/etc/utimaco`. Locate the Utimaco PKCS#11 configuration file in your SecurityServer directory, `Software\Linux\Crypto_APIs\PKCS11_R3\sample`. Copy the Utimaco PKCS#11 configuration file `cs_pkcs11_R3.cfg` to the `/etc/utimaco` directory.

```
# mkdir /etc/utimaco
# cd ~/path_to_application_folder/Software/Linux/Crypto_APIs/PKCS11_R3/sample
# cp cs_pkcs11_R3.cfg /etc/utimaco
```

On Windows:

On Windows, `cs_pkcs11_R3.cfg` will be automatically created and available in the `C:\ProgramData\UtimacoPKCS11_R3` folder as part of the SecurityServer software installation. Edit the `cs_pkcs11_R3.cfg` file and make the appropriate changes to it. A sample `cs_pkcs11_R3.cfg` file is mentioned below:

```
library = C:\oracle\extapi\64\hsm\utimaco\6.1.1.0\cs_pkcs11_R3.dll
 slot = 0
 pin = Oracle123
 [Global]
 # For Unix:
 Logpath = /tmp
 # For Windows:
 # Logpath = C:/ProgramData/Utimaco/PKCS11_R3
 # Loglevel (0 = NONE; 1 = ERROR; 2 = WARNING; 3 = INFO; 4 = TRACE)
 Logging = 1
 # Prevents expiring session after inactivity of 15 minutes
 KeepAlive = true
 # Set the Device to connect with
 #[CryptoServer]
 # Device specifier
 Device = <HSM_IP>
```

For detailed guidance on commands and their parameters, please refer to the Utimaco SecurityServer documentation. The device could be a u.trust GP HSM Se-Series, available in either PCIe or LAN form factors. Depending on the type, the device configuration line will follow one of these formats:

- **LAN-based HSM:** Device = 288@ipaddress
- **PCIe-based HSM:** Device = /dev/cs2.0

Select the appropriate format based on your specific hardware setup.

`library` specifies the path where the `cs_pkcs11_R3.dll` file is located.

`Slot` indicates the slot number associated with the created USER.

`Pin` represents the password assigned to the USER.

To simplify your testing process, it's recommended that you enable the PKCS#11 log file by adjusting the logging settings. Specifically:

- Set the `LogPath` to a writable directory (not a specific file).
- Set the `Logging` log level to 1 for basic logging. Increase it to 4 for more detailed output during testing.

This will generate a log file named `cs_pkcs11_R3.log` within the specified `LogPath` directory. Reviewing this log can help with troubleshooting if you encounter issues. Once testing is complete, it's advisable to reduce `Logging` log level to 1 or 2 to limit output to only critical or important messages.

# 5 Installing and Configuring JSign

## 5.1 Install JSign

1. Download the latest JSign JAR(7.4) from the official releases page https://github.com/ebourg/jsign/releases.
2. Create a directory `C:\Tools\jsign` and copy the downloaded JAR file into that folder.
3. Open a PowerShell window in admin mode and open the `jsign` folder.
4. Check the `jsign` version.

```
cd C:\Tools\jsign
java -jar jsign-7.4.jar --version
```

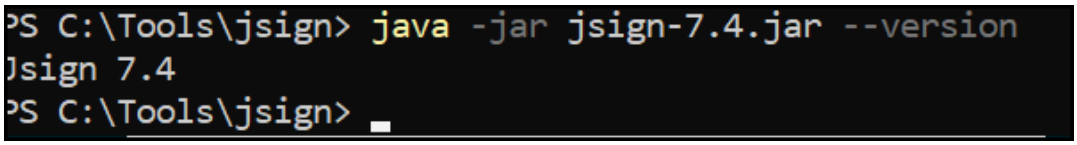


Figure 1 : JSign version detail

## 5.2 Install the PowerShell PKI Module

PKI Module is intended to simplify various PKI management tasks by using automation with Windows PowerShell. It is intended for Certification Authority (CA) management.
1. Install all the PKI modules using the below command in PowerShell.

```
C:\> Install-Module -Name PSPKI
```

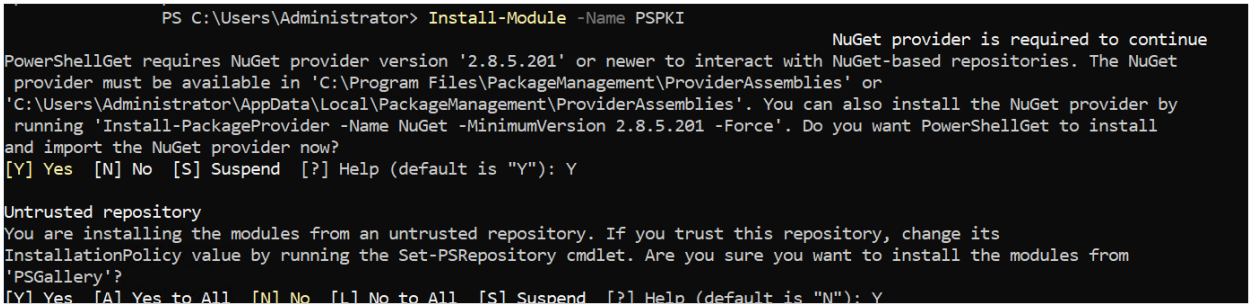


Figure 2 : Install PKI module

2. Verify the following PKI modules are installed correctly by running the following command.

```
Get-Command -Module PKI
```

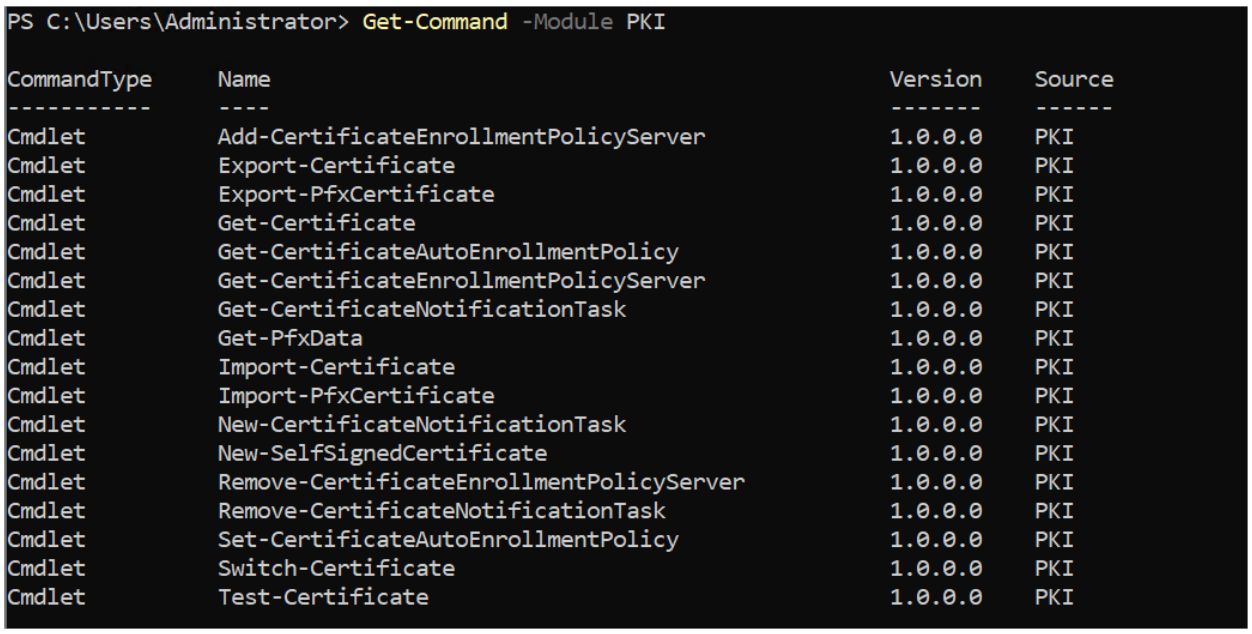


Figure 3 : PKI module details

# 6 Integration Steps on Utimaco u.trust GP HSM Se-Series

The following section outlines the procedures required to configure both the Utimaco HSM and JSign components for seamless integration.

Create users SO (Security Officer) and USR (the Crypto user) accounts and initialize a slot.

The slot must be initialized using the p11tool2.

First, create the SO using p11tool2. Then, using the `p11tool2` command, initialize the slot you want to use and the slot user, as shown below.

- `# ./p11tool2 slot=<slot no.> Label=<token label> Login=ADMIN,ADMIN.key InitToken=<SO pin>`

Initialize the SO user.

- `# ./p11tool2 slot=<slot no.> LoginSO=<SO pin> InitPin=<Cryptouser pin>`

Ensure that the Utimaco u.trust GP HSM is reachable and accessible from the system where JSign is installed.

# 7 Integration Steps on JSign

## 7.1 Generate a Signing Certificate

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

```
p11tool2.exe Slot=0 LoginUser=<passcode>
PubKeyAttr='CKA_LABEL="jk_public_key",CKA_MODULUS_BITS=2048,CKA_ID="01"'
 PrvKeyAttr='CKA_LABEL="jk_private_key",CKA_EXTRACTABLE=CK_TRUE,CKA_ID="01"'
 GenerateKeyPair=RSA
```

2. Verify that the keys are generated.

```
p11tool2.exe LoginUser=<passcode> ListObjects
```

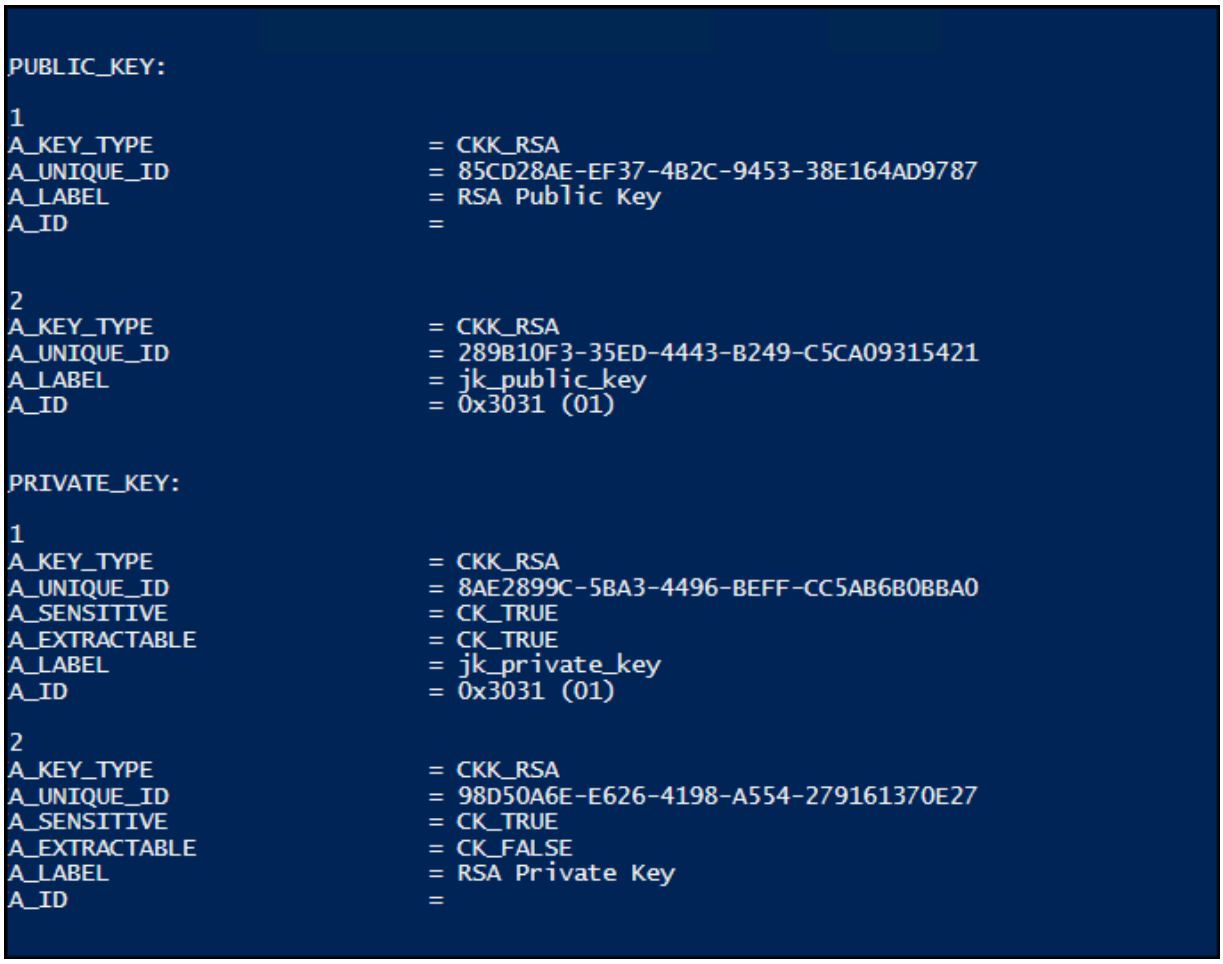


Figure 4 : Generated keys displayed

3. Generate a certificate signing request using the HSM-based private key.

```
p11tool2.exe Slot=0 LoginUser=<passcode>
PubKeyAttr=CKA_LABEL="jk_public_key",CKA_MODULUS_BITS=2048,CKA_ID="01"
PrvKeyAttr=CKA_LABEL="jk_private_key",CKA_EXTRACTABLE=CK_TRUE,CKA_ID="01"
Mech=CKM_SHA256_RSA_PKCS DN="O=Utimaco,CN=Utimaco JCode Signing"
ExportP10=testhsm_js.csr

openssl req -in "<path>\testhsm_js.csr" -text -noout
```

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=Utimaco,CN=Utimaco JCode Signing"
Defines the subject name for the CSR:
O → Organization name (Utimaco)
CN → Common Name (name associated with the code-signing certificate subject)

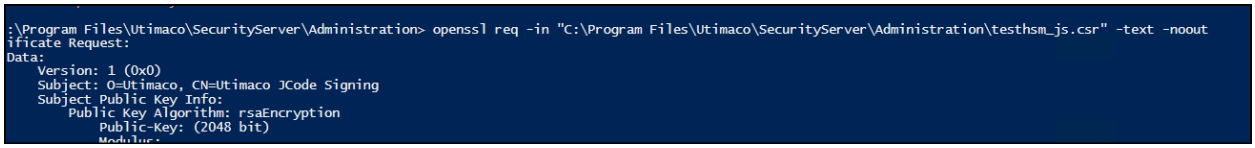


Figure 5 : CSR generated

4. Submit the generated Certificate Signing Request (CSR) to a trusted Public Certificate Authority (CA) to obtain a signed certificate. Store the signed certificate in JSign environment.

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

5. Convert the signed certificate to an HSM supported format and import to HSM.

```
 openssl x509 -in "<path>\testhsm_js.cer" -outform DER -out
"<path>\testhsm_js.der"

 p11tool2.exe Slot=0 LoginUser=<passcode>
CertAttr='CKA_LABEL="jk_private_key",CKA_ID="01"'
ImportCert="<path>\testhsm_js.der"
```

6. Verify the imported certificate available in HSM.

```
p11tool2.exe LoginUser=<password> ListObjects
```

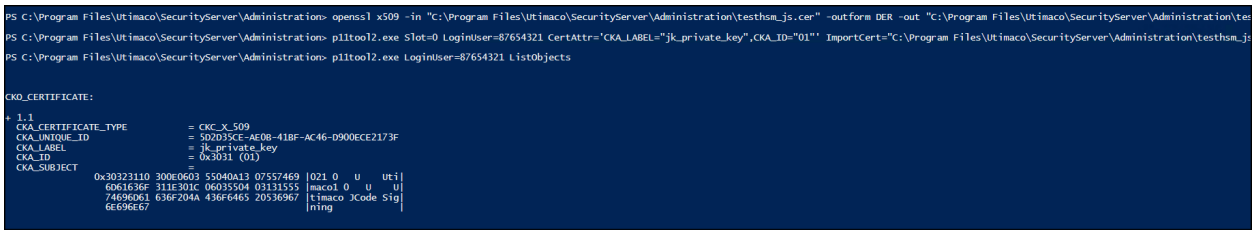


Figure 6 : Imported certificate displayed in HSM

## 7.2 Create a Self-Signed Certificate

1. Use the following Java `keytool` command to generate an RSA key and certificate through the Utimaco PKCS#11 provider. The private key and self-signed certificate will be stored in the HSM.

```
keytool -genkeypair -alias <key label> -keystore NONE -storetype PKCS11 -storepass
<passcode> -keyalg RSA -dname "CN=Utimaco Code Signing,O=Utimaco,L=Aachen,C=DE"
-providerClass sun.security.pkcs11.SunPKCS11 -providerArg "C:\Tools\jsign\utimaco-
pkcs11-java.cfg"
```

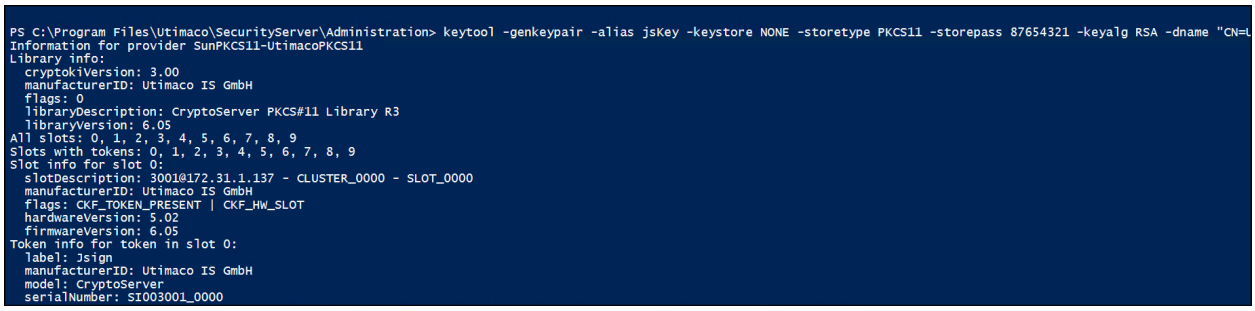


Figure 7 : Self-signed certificate created

2. Use the command below to connect Java Keytool directly to the HSM via PKCS#11, and list the keys and certificates available in the token. Upon successful execution get a response as “Your keystore contains <n> entry”.

```
keytool -list -keystore NONE -storetype PKCS11 -storepass <passcode>
-providerClass sun.security.pkcs11.SunPKCS11 -providerArg "C:\Tools\jsign\utimaco-
pkcs11-java.cfg"
```

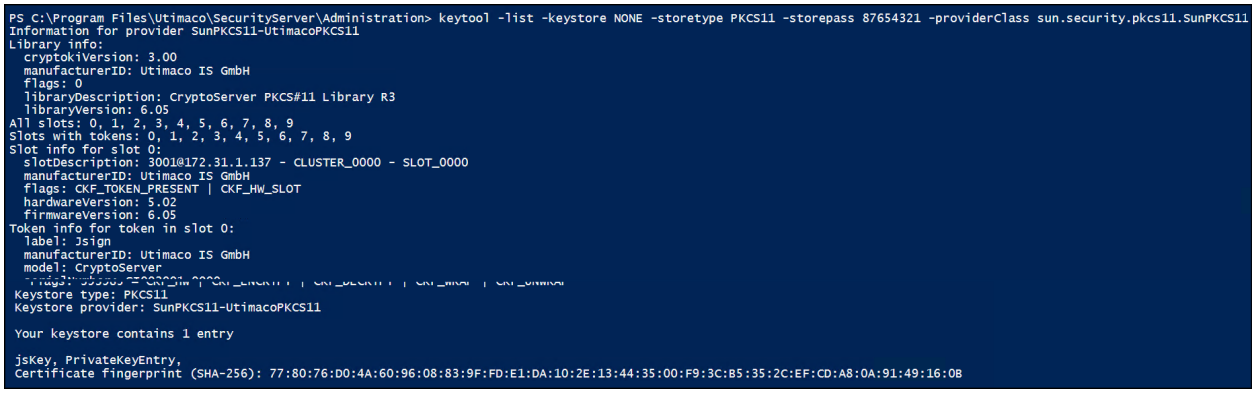


Figure 8 : Key and certificate available in HSM

3. Verify the key and certificate available in HSM.

```
p11tool2.exe LoginUser=<passcode> ListObjects
```

```
PS C:\Program Files\Utimaco\SecurityServer\Administration> p11tool2.exe LoginUser=87654321 ListObjects


CKO_CERTIFICATE:

+ 1.1
  CKA_CERTIFICATE_TYPE          = CKC_X_509
  CKA_UNIQUE_ID                 = 08D00692-800C-4020-8DB0-42AFE83997D7
  CKA_LABEL                     = jsKey
  CKA_ID                        = 0x6A734B65 79 (jsKey)
  CKA_SUBJECT                   =
                0x304F310B 30090603 55040613 02444531 |0O1 0   U    DE1|
                  0F300D06 03550407 13064161 6368656E | 0   U    Aachen|
                  3110300E 06035504 0A130755 74696D61 |1 0   U    Utima|
                  636F311D 301B0603 55040313 14557469 |co1 0   U    Uti|
                  6D61636F 20436F64 65205369 676E696E |maco Code Signin|
                  67                                  |g               |


CKO_PUBLIC_KEY:

+ 2.1
  CKA_KEY_TYPE                  = CKK_RSA
  CKA_UNIQUE_ID                 = 85CD28AE-EF37-4B2C-9453-38E164AD9787
  CKA_LABEL                     = RSA Public Key
  CKA_ID                        =


CKO_PRIVATE_KEY:

+ 3.1
  CKA_KEY_TYPE                  = CKK_RSA
  CKA_UNIQUE_ID                 = 98D50A6E-E626-4198-A554-279161370E27
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     = RSA Private Key
  CKA_ID                        =

+ 3.2
  CKA_KEY_TYPE                  = CKK_RSA
  CKA_UNIQUE_ID                 = 022090C8-DCC2-4B4C-9900-EBB839765448
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     =
  CKA_ID                        = 0x6A734B65 79 (jsKey)
```

Figure 9 : Key and certificate details in HSM

# 8 Verification and Testing

## 8.1 Code Signing

Once the signing certificate has been imported to the HSM, executables, dynamic-link libraries (DLLs), and cabinet (CAB) files can be digitally signed. Use the following PowerShell command to sign the application. In this example, `TestSign.exe` is used as a test application to validate the signing process.

```
 java -jar "C:\Tools\jsign\jsign-7.4.jar" --keystore "C:\Tools\jsign\utimaco-
pkcs11-java.cfg" --storetype PKCS11 --storepass <passcode> --alias jsKey --tsaurl
http://timestamp.digicert.com "C:\Tools\jsign\TestSign.exe"
```


Figure 10 : Code-signing

If the code-signing operation is successful, the message "Adding Authenticode signature" is displayed, indicating that the digital signature has been successfully applied to the file.

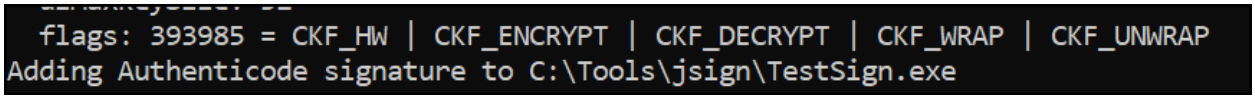


Figure 11 : Code-signing completed

## 8.2 Verification

1. Using the below command verify that the application is now signed.

```
Get-AuthenticodeSignature "<executable file path>\<application name>.exe"
```

```
PS C:\Program Files\Utimaco\SecurityServer\Administration> Get-AuthenticodeSignature "C:\Tools\jsign\TestSign.exe"

    Directory: C:\Tools\jsign

SignerCertificate                         Status                                                  Path
-----------------                         ------                                                  ----
312860D2047EB81F8F58C29FF19ECDB4C634CF6A  Valid                                                   TestSign.exe
```

Figure 12 : Valid certificate details

2. Verify that the application is now signed by right-clicking on it and selecting **Properties**. On the **Digital Signatures tab** (if it exists), you can view the signing certificate and timestamp.

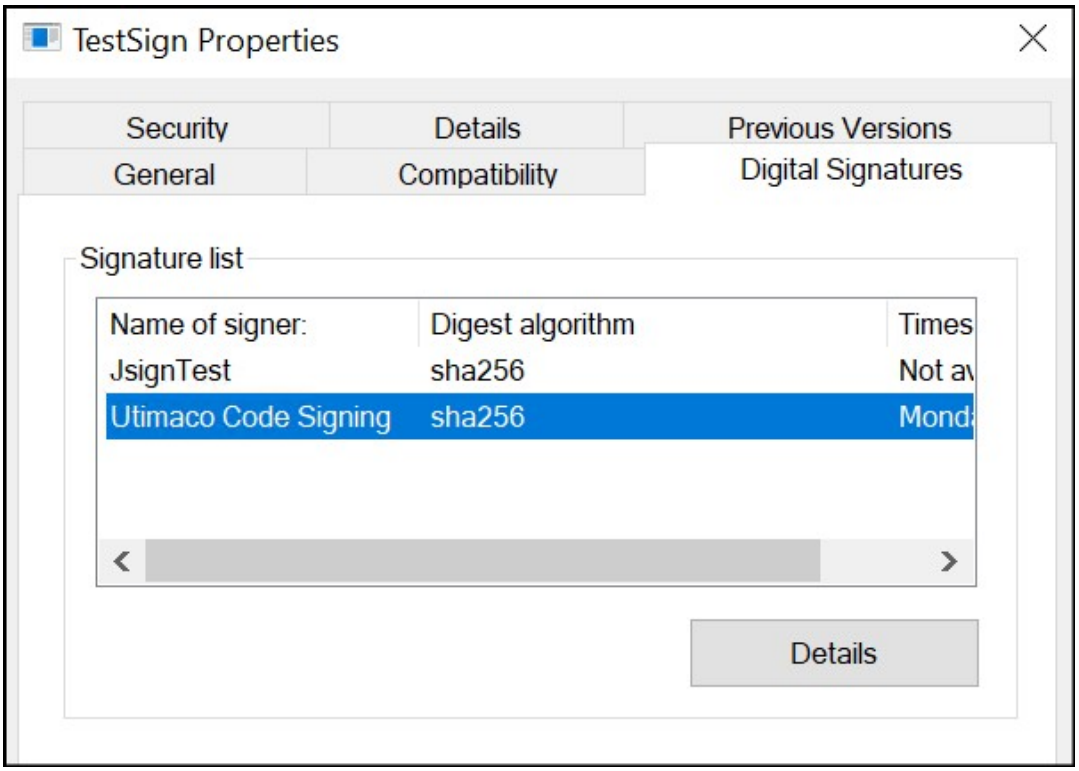


Figure 13 : Signing certificate details

# 9 Troubleshooting

## 9.1 Log locations and interpretation

PKCS#11 Logs:

Enabling PKCS#11 logging to facilitate easier testing and troubleshooting is recommended. This can be done by configuring the Logging Loglevel and LogPath parameters in the configuration file.

- LogPath should point to a writable directory (not a specific file) where log files can be stored.
- Logging Loglevel controls the verbosity of the logs:
  - Set it to 1 for basic logging.
  - For detailed testing and debugging, increase the level to 4.

The generated log file will be named `cs_pkcs11_R3.log` and located in the directory specified by LogPath. Reviewing this log file can help identify and resolve issues that arise during testing.

Once testing is complete, it is advisable to reduce the Logging Loglevel to 1 or 2 to limit logging to only critical or important messages, thereby optimizing performance and reducing unnecessary log data.

# 10 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.

# 11 Appendices

## 11.1 References

| Title | Description | Document/Link |
|---|---|---|
| [CSADMIN] | CryptoServer – csadm Manual/ Utimaco IS GmbH | 2009-0003 |
| [CSADMIN2] | CryptoServer_csadm_Manual_Systemadministrators.pdf | 2009-0003 |
| [CSLAN] | CryptoServerLAN_V5_Manual_Systemadministrators.pdf | 2018-0010 |
| [CSP-CNG] | CryptoServer_Manual_CSP_CNG.pdf | 2008-0002 |
| JSign Releases | JSign latest release details. | *https://github.com/ebourg/jsign/releases* |
| JSign Documentation | JSign documentation details. | *https://ebourg.github.io/jsign* |

Table 6: References

## 11.2 Command Summary (CLI commands used)

| Commands Used | Purpose |
|---|---|
| `java -jar jsign-7.4.jar --version` | Verify the installed JSign version. |
| `Install-Module -Name PSPKI` | Install the PowerShell PKI module. |

| Commands Used | Purpose |
|---|---|
| `Get-Command -Module PKI` | Verify PKI module installation. |
| `./p11tool2 slot=<slot no.> Label=<token label> Login=ADMIN,ADMIN.key InitToken=<SO pin>` | Create and initialize a PKCS#11 token. |
| `./p11tool2 slot=<slot no.> LoginSO=<SO pin> InitPin=<Crypto user pin>` | Initialize the Crypto User PIN. |
| `p11tool2.exe Slot=0 LoginUser=<passcode> PubKeyAttr='CKA_LABEL="jk_public_key",CKA_MODULUS_BITS=2048,CKA_ID="01"' PrvKeyAttr='CKA_LABEL="jk_private_key",CKA_EXTRACTABLE=CK_TRUE,CKA_ID="01"' GenerateKeyPair=RSA` | Generate RSA key pair in HSM. |
| `p11tool2.exe LoginUser=<passcode> ListObjects` | List HSM objects and verify key/ certificate creation. |
| `p11tool2.exe Slot=0 LoginUser=<passcode> PubKeyAttr=CKA_LABEL="jk_public_key",CKA_MODULUS_BITS=2048,CKA_ID="01" PrvKeyAttr=CKA_LABEL="jk_private_key",CKA_EXTRACTABLE=CK_TRUE,CKA_ID="01" Mech=CKM_SHA256_RSA_PKCS DN="O=Utimaco,CN=Utimaco JCode Signing" ExportP10=testhsm_js.csr` | Generate a CSR using an HSM-protected private key. |
| `openssl req -in "<path>\testhsm_js.csr" -text -noout` | Verify CSR content. |

| Commands Used | Purpose |
|---|---|
| `openssl x509 -in "<path>\testhsm_js.cer" -outform DER -out "<path>\testhsm_js.der"` | Convert signed certificate to DER format. |
| `p11tool2.exe Slot=0 LoginUser=<passcode> CertAttr='CKA_LABEL="jk_private_key",CKA_ID="01"' ImportCert="<path>\testhsm_js.der"` | Import signed certificate into HSM. |
| `keytool -genkeypair -alias <key label> -keystore NONE -storetype PKCS11 -storepass <passcode> -keyalg RSA -dname "CN=Utimaco Code Signing,O=Utimaco,L=Aachen,C=DE" -providerClass sun.security.pkcs11.SunPKCS11 -providerArg "C:\Tools\jsign\utimaco-pkcs11-java.cfg"` | Generate self-signed certificate directly in HSM. |
| `keytool -list -keystore NONE -storetype PKCS11 -storepass <passcode> -providerClass sun.security.pkcs11.SunPKCS11 -providerArg "C:\Tools\jsign\utimaco-pkcs11-java.cfg"` | Verify PrivateKeyEntry and certificate availability. |
| `java -jar "C:\Tools\jsign\jsign-7.4.jar" --keystore "C:\Tools\jsign\utimaco-pkcs11-java.cfg" --storetype PKCS11 --storepass <passcode> --alias <alias> --tsaurl http://timestamp.digicert.com "C:\Tools\jsign\TestSign.exe"` | Digitally sign an executable using the HSM key. |
| `Get-AuthenticodeSignature "<executable file>"` | Verify Authenticode signature. |

Table 7: Command summary