

OpenStack Barbican Dalmatian

Integration Guide

CP5

v5.2.0.11

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	1.1.0
Date	2026-02-26
Status	PUBLISHED
Document No.	IG-2026-0024
All rights reserved	<p>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.</p> <p>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.</p> <p>All trademarks and registered trademarks are the property of their respective owners.</p>

Table of Contents

1	Introduction	5
1.1	About This Guide	5
1.2	Target Audience for This Guide	5
1.3	Document Conventions	5
1.4	Abbreviations	6
2	Product Overview	8
2.1	OpenStack Barbican	8
2.2	Utimaco CP5 [VS-NfD]	8
3	Integration Requirements and Prerequisites	9
3.1	Tested Versions	9
3.2	Software Requirements	9
3.3	Hardware Requirements	10
3.4	Prerequisites	10
4	Installation and Configuration	11
4.1	Download and Install Utimaco CryptoServer Software	11
4.2	CryptoServer PKCS#11 Configuration	12
4.3	(Optional) Installing OpenStack Barbican	14
5	Integrating OpenStack Barbican with Utimaco SecurityServer	17
5.1	Configuration on Utimaco CryptoServer GP HSM	17
5.2	Configure OpenStack Barbican to Use Utimaco HSM	19
5.3	Generating MKEK and HMAC Key on Utimaco HSM	20
5.4	Initialize and Authorize HMAC and MKEK for CP5	22
6	Verification and Testing	28
6.1	Functional Testing	28
6.1.1	Encrypting and Decrypting Secrets	28
6.1.2	Generating a Symmetric Key in OpenStack Barbican	30
6.1.3	Storing Public Key, Private Key and Certificate in OpenStack Barbican	32
6.1.4	Key Rotation/Migration	39
7	Troubleshooting	44
7.1	Common Issues and How To Resolve Them	44
7.2	Log Location and Interpretation	45

7.2.1	PKCS#11 Logs.....	45
7.2.2	OpenStack Barbican Logs	45
8	Further Information	47
8.1	References	47

1 Introduction

This guide is part of the information and support provided by Utimaco. Additional documentation produced to support your Utimaco SecurityServer product can be found in the document directory of the Utimaco SecurityServer product bundle. All Utimaco SecurityServer product documentation is available from Utimaco's website at <https://utimaco.com/>.

1.1 About This Guide

This guide describes how to enable HSM integration with OpenStack Barbican Dalmatian. Utimaco HSM securely stores the MKEK and HMAC used by Barbican.

1.2 Target Audience for This Guide

This guide is intended for OpenStack Barbican and Utimaco Hardware Security Module (HSM) administrators.

1.3 Document Conventions

The following conventions are used in this guide:

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

Table 1: Document conventions

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



Here you find important safety information that should be followed.



Here you find additional notes or supplementary information.



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

1.4 Abbreviations

The following abbreviations are used in this guide:

Abbreviation	Meaning
API	Application Programming Interface
CSADM	CryptoServer Command-line Administration Tool
CSP	Cryptographic Service Provider
CXI	Cryptographic eXtended Services Interface
DB	Database
GUI	Graphical User Interface
HMAC	Hash-based message authentication code

HSM	Hardware Security Module
IP	Internet Protocol
LAN	Local Area Network
MBK	Master Backup Key
MKEK	Master Key Encryption Key
PCIe	PCI Express Interface
URL	Uniform Resource Locator

Table 2: Abbreviations

2 Product Overview

2.1 OpenStack Barbican

Barbican is the OpenStack Key Manager service. It provides secure storage, provisioning, and management of secret data. This includes keying material such as Symmetric Keys, Asymmetric Keys, Certificates, and raw binary data.

2.2 Utimaco CP5 [VS-NfD]

Utimaco CryptoServer CP5 [VS-NfD] is a hardware security module that is suitable for a variety of use cases, such as certificate issuing for signature creation or authentication, and signing of classified documents. The German Federal Office for Information Security (BSI) has approved the VS-NfD compliance version on CryptoServer General Purpose HSM according to the Qualified Certification Procedure – the product is the only VS-NfD hardware security module approved by the BSI. It allows to secure data classified up to “VS-NfD”, “EU restricted” and “NATO restricted”.

3 Integration Requirements and Prerequisites

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

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

3.1 Tested Versions

The integrations that have been successfully tested with the Utimaco HSM and OpenStack Barbican:

Operating System	OpenStack Version	Utimaco Crypto Server Version	Utimaco HSM
Ubuntu 22.04	OpenStack Dalmatian	CryptoServerCP5-NfD-V5.2.0.11	CryptoServer General Purpose HSM

Table 3: List of tested versions

3.2 Software Requirements

Software	Software Requirements
HSM Interface	CryptoServer CP5 PKCS#11 Provider
HSM Utility	CryptoServer CP5 PKCS#11 Tool (p11tool2)

Table 4: List of software requirements

3.3 Hardware Requirements

Hardware	Hardware Requirements
Utimaco LAN HSM	CryptoServer General Purpose HSM LAN with CryptoServerCP5-NfD-V5.2.0.1 firmware
Utimaco PCI-e HSM	CryptoServer General Purpose HSM PCIe with CryptoServerCP5-NfD-V5.2.0.1 firmware

Table 5: List of hardware requirements

3.4 Prerequisites

Before you begin, please ensure that you have:

- Installed and set up the operating system listed in [Tested Versions](#).
- Installed and set up the HSM listed in [Tested Versions](#).
- Replaced the HSM default admin with a new admin user.
- Created and stored the MBK on each HSM. Refer to the CryptoServer documentation to set up the MBK.
- Set up and configured the CryptoServer. Refer to the CryptoServer documentation to set up the HSM.
- An admin user, which is required to install software.

4 Installation and Configuration

4.1 Download and Install Utimaco CryptoServer Software

If you have not already done so, create and request an Utimaco Support Portal Account at [Support - Utimaco Portal](#). This will allow you to download the software components needed for this installation.



The PKCS#11 software is not included in the CryptoServerCP5-NfD-V5.2.0.1 package due to certification reasons. VS-NfD SupportingCD will also need to be downloaded as it includes the missing software.

1. Copy the downloaded software to the appropriate location on the OpenStack Barbican Server.
2. Create `/utimaco/bin` and `/utimaco/lib` directories under `/opt`.

```
>_ Console
```

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

3. Copy PKCS#11 library file `libcs_pkcs11_R2.so` from the SupportingCD package to the `/opt/utimaco/lib` directory and make the file executable.

```
>_ Console
```

```
# cp ~/SupportingCD-V5.2.0.1/Software/Linux/x86_64/CryptoAPIs/PKCS11_R2/  
lib/cs_pkcs11_R2.so /opt/utimaco/lib  
# chmod +x /opt/utimaco/lib/cs_pkcs11_R2.so
```

4. Copy the csadm and cxitool files from the Utimaco CryptoServerCP5[VS-NfD] package and the p11tool2 from the SupportingCD package to `/opt/utimaco/bin` directory and make all files executable.

```
>_ Console
```

```
# cd ~/CryptoServerCP5-NfD-  
V5.2.0.1.zip\Software\Linux\x86-64\Administration  
  
# cp csadm /opt/utimaco/bin  
  
# cd ~/CryptoServerCP5-NfD-  
V5.2.0.1.zip\Software\Linux\x86-64\Crypto_APIs\CXI\bin  
# cp cxitool /opt/utimaco/bin  
  
# cd ~/SupportingCD-V5.2.0.1/Software/Linux/x86_64/CryptoAPIs/PKCS11_R2/  
bin  
  
# cp p11tool2 /opt/utimaco/bin  
  
# chmod +x /opt/utimaco/bin/csadm /opt/utimaco/bin/p11tool2 /opt/utimaco/  
bin/cxitool
```

4.2 CryptoServer PKCS#11 Configuration

1. Create the directory `/etc/utimaco`. Locate the Utimaco PKCS#11 configuration file in your SupportingCD directory (`Linux/x86-64/Crypto_APIS/PKCS11_R2/sample`). Copy the Utimaco PKCS#11 configuration file `cs_pkcs11_R2.cfg` into `/etc/utimaco` directory.

```
>_ Console
```

```
# mkdir /etc/utimaco

# cd ~/~/SupportingCD-V5.2.0.1.zip/Software/Linux/x86_64/CryptoAPIs/
PKCS11_R2/sample

# cp cs_pkcs11_R2.cfg /etc/utimaco

# chmod +x cs_pkcs_R2.cfg
```

2. Edit the `cs_pkcs11_R2.cfg` file and make the appropriate changes to the file.

```
cs_pkcs11_R2.cfg
```

```
[Global]

# For unix:

Logpath = /tmp

# LogLevel (0 = NONE; 1 = ERROR; 2 = WARNING; 3 = INFO; 4 = TRACE)

Logging = 4

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 CryptoServer documentation. The device could be a CryptoServer GP HSM, 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

Make sure to select the appropriate format based on your specific hardware setup.



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` 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_R2.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` level to 1 or 2 to limit output to only critical or important messages.

4.3 (Optional) Installing OpenStack Barbican



Skip this section if OpenStack Barbican is already installed. The below steps are only for demonstration purposes and will change based on actual requirements.

Use the steps below to install OpenStack Dalmatian with Barbican via DevStack on a single node. DevStack is a series of extensible scripts used to quickly bring up a complete OpenStack environment either based on the latest versions of everything from git master or a selected release (Barbican, Flamingo, etc.).

1. Create `stack` user with passwordless sudo privileges.

```
>_ Console
```

```
# sudo useradd -s /bin/bash -d /opt/stack -m stack
# echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
# sudo chmod 0440 /etc/sudoers.d/stack
# sudo su - stack
```

2. Clone the DevStack Repository and switch to stable Dalmatian 2024.2 release branch.

```
>_ Console
```

```
# cd /opt/stack
# git clone https://opendev.org/openstack/devstack
# cd devstack
# git checkout stable/2024.2
```

3. Create the DevStack configuration file `local.conf` with the following content.

```
>_ Console
```

```
cat > local.conf << 'EOF'
[[local|localrc]]

ADMIN_PASSWORD=secret
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD

enable_plugin barbican https://opendev.org/openstack/barbican stable/
2024.2

HOST_IP=127.0.0.1

LOGFILE=$DEST/logs/stack.sh.log

enable_service barbican
EOF
```

Adjust `HOST_IP` to your machine's IP address if accessing from other systems. For single-machine testing, `127.0.0.1` is sufficient.

4. Run DevStack installation.

```
>_ Console
```

```
# ./stack.sh
```

This process typically takes 15-30 minutes depending on your network speed and system resources. The script will download, configure, and start all necessary OpenStack services including Barbican.

5. Verify the installation by sourcing the OpenStack credentials and then listing the Barbican secrets, which should return empty initially and check the Barbican service status.

```
>_ Console
```

```
# source ~/devstack/openrc admin admin
```

```
# openstack secret list
```

```
# systemctl status devstack@barbican-*.service
```

5 Integrating OpenStack Barbican with Utimaco SecurityServer

5.1 Configuration on Utimaco CryptoServer GP HSM

1. Create users SO (Security Officer) and USR (the Crypto user) and initialize a slot. This is done using p11tool2. We first create the SO using the `InitToken` command and then the USR using the `InitPin` command.

```
>_ Console
```

```
# ./p11tool2 Slot=<slot no.> Label=BarbicanToken Login=ADMIN,ADMIN_CP5.key  
InitToken=<SO pin>
```

```
# ./p11tool2 Slot=<slot no.> LoginSO=<SO_pin> InitPin=<CryptoUser_PIN>
```

2. Run the `csadm ListUsers` command to see the created users.

```
root@ub22-openstack-barbican:/opt/utimaco/bin# ./csadm Dev=3001@localhost ListUsers  
Name      Permission  Mechanism  Attributes  
ADMIN     22000000   RSA sign   Z[0]  
ADMIN2    22000000   RSA sign   Z[0]  
SO_0001   00000200   HMAC passwd Z[1]A[CXI_GROUP=SL0T_0001]L[Barbican  
USR_0001  00000022   HMAC passwd Z[0]A[CXI_GROUP=SL0T_0001]
```

Figure 1 : Security officer and cryptographic user created

3. Export the HSMs mutual authentication key. The `port@address` is `3001@localhost` if running the simulator locally. Hardware HSM will have the port 288 (`288@address`).

```
>_ Console
```

```
# cd /opt/Utimaco/bin

# ./csadm dev=<port@address> GetHSMAuthKey > /opt/utimaco/HSMAuthKey.key

# cat /opt/utimaco/HSMAuthKey.key

CS000000=0601020180000180A9611F90A8377FDFC90BC3097556F5873EFE31F9F697E44AA7
C81A91FD9154D84EDDCDF53340E8891D7E79F445ECEFFEE1FBF6C014E054E1787C43816A6
C8D1E1C75C44F534F890B57CF3A876CC47C68C5CE8D5C0EA05D9E6FBED4AD7C2905BC2527D8
32C67CE298B7946B0A10FA23048A74D91D1EBA487DFDDAF4AE0B9EA25AC3213EB7874AE8700
A18D36A20C9A915A2A1216DEEBA872FB296DDF28587ADF805CDEC7A63178C2F25956AB1B903
7D4A8774273E6A3CBBF5CA90221EAC91211AB1E848DB0A26A4B94AF5F835B8DA0B01E17759C
C0A746FA834FEBAE24604228C7AF8442FA651FC9ED8545C23D5BF313D3DBD9D662B9B3162A4
D42CC4B3F4F3E73B57475836B7A89A228E3F9793A488031E2D09AB48F06ABEEA18618CDB6BF
FE7C2D370F7C1AEF3C1B03E53D73E45CE58B3CEB81C514FFA691FBA087D6C24E31DC17123B3
B4DF2880CE44E278DBF8A23B35A7FEC9228808F560F062DA984B0A65B6ACB7AB1A0EC716A1F
31268039CA4DC37B4DDBC989AF4C15723B8364758D95554494D41434F204353303030303030
018099DEC18B5179B5A8FE0F88AA24D57A272584A4221B591AC3D513093763597E2A5F153D7
2AD896C24A57509D5FC21D8F1ABD621825A18B2D9497857C5169C0A2286CA334986113A23A5
B3D786C711824F98C764AD0C3AF6A45FE0823363BE3D7ACD2CD0308BD4EB36879136F0A12BD
D8C13EE58869DFAA24596C4BDA8436D3BE259F5E25A0EC3D1E3CF38F92969D7A65F709049B9
5DB3546E5740EAA28F0F8D960AE4A243C6380C091D7BC1773C50EC88B1CC4D9705D99DD8DA4
D7164903F7A9B76BB73C488B003F41BE2A1A4DF3C13229DD69D64E7B50D7C6624BAA2611E81
98990BA502D33CF21472A57659E8B728FAC25B8B9113921854878802C00BA083EDC0D910CD9
3655ED2004010D38CCE6B43D72564559F7CBA7212EE28EA5DF63174B937918BC65365046F5C
9DCF4121A3ED711E3D037C6655BD99939B4BF80F795CB0F755DCCD2627225CF6C76ADC61580
EA98EE32ADE63E518C032BC36A44B939BD0C035573FDF1110B3B701883CD8FD2D8D8D36E882
BB6292DDC40A54F2A5B5CD

#export CS_AUTH_KEYS="/opt/utimaco/HSMAuthKey.key"
```

You can also add the above variable in users .bash_profile to make it persistent through multiple session.

```
>_ Console
```

```
# echo 'export CS_AUTH_KEYS="/opt/utimaco/HSMAuthKey.key"' >> ~/.bashrc
```

5.2 Configure OpenStack Barbican to Use Utimaco HSM

1. Add the below information to the `barbican.conf` file. Make sure to replace the placeholders in <> with relevant values.

`barbican.conf`

```
[secretstore]
namespace = barbican.secretstore.plugin
enabled_secretstore_plugins = store_crypto

[crypto]
enabled_crypto_plugins = p11_crypto

[p11_crypto_plugin]

# Path to Utimaco PKCS#11 library
library_path = /opt/utimaco/lib/libcs_pkcs11_R2.so

# CryptoUser PIN to login to PKCS#11
login = <CryptoUser_PIN>

# Master Key Encryption Key and HMAC labels
mkek_label = mkek_utimaco
mkek_length = 32
hmac_label = hmac_utimaco

# HSM Slot ID (Integer value)
slot_id = <slot_id>

# AES encryption mechanism (as used by Utimaco)
encryption_mechanism = CKM_AES_CBC
```



`mkek_utimaco` and `hmac_utimaco` keys will be generated on the Utimaco HSM in slot 1 in the next section of this document.

5.3 Generating MKEK and HMAC Key on Utimaco HSM

1. Generate the MKEK using the `p11tool2 GenerateKey` command below.

```
>_ Console

# ./p11tool2 Slot=<slot_id> LoginUser=<CryptoUser_PIN>
KeyAttr=CKA_LABEL="mkek_utimaco",CKA_VALUE_LEN=32,CKA_WRAP=true,CKA_UNWRAP=
true,CKA_ENCRYPT=false,CKA_DECRYPT=false,CKA_EXTRACTABLE=false
GenerateKey=AES
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./p11tool2 Slot=1 LoginUser=12345678 KeyAttr=CKA_LABEL="mkek_utimaco",CKA_VALUE_LEN=32,CKA_WRAP=true,CKA_UNWRAP=true,
CKA_ENCRYPT=false,CKA_DECRYPT=false,CKA_EXTRACTABLE=false GenerateKey=AES
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./p11tool2 Slot=1 LoginUser=12345678 ListObjects

CKO_SECRET_KEY:
+ 1.1
  CKA_KEY_TYPE           = CKK_AES
  CKA_SENSITIVE          = CK_TRUE
  CKA_EXTRACTABLE        = CK_FALSE
  CKA_LABEL               = mkek_utimaco
  CKA_ID                 =
  CKA_UTC_AUTH_COUNTER   = 0
```

Figure 2 : MKEK generation



Barbican can be used to generate an MKEK. However, using `barbican-manage hsm gen_mkek` causes a `key property invalid in CC mode` error when attempting to create the key. The reason for that is that when HSM is operated in CC mode (which is the case for CryptoServer CP5), a key can not be used for both encryption/decryption and wrapping/unwrapping. The command adds the `CKA_ENCRYPT` and `CKA_DECRYPT` usage flags to the key. To work around this issue, the key is created with custom attributes using the `p11tool2` with `CKA_ENCRYPT` and `CKY_DECRYPT` set to `false` as the key is only required for wrapping/unwrapping.

2. Generate the HMAC using the `barbican-manage hsm gen_hmac` command.

```
>_ Console
```

```
# sudo -u stack -E barbican-manage hsm gen_hmac --library-path '/opt/utimaco/lib/libcs_pkcs11_R2.so' --passphrase <CryptoUser_PIN> --slot-id <slot_id> --label 'hmac_utimaco' --length 32
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ sudo -u stack -E barbican-manage hsm gen_hmac --library-path '/opt/utimaco/lib/libcs_pkcs11_R2.so' --passphrase 12345678 --slot-id 1 --label 'hmac_utimaco' --length 32
HMAC successfully generated!
```

Figure 3 : HMAC generation

3. Verify that the keys are generated on the Utimaco HSM using the `p11tool2` `ListObjects` command.

```
>_ Console
```

```
#!/p11tool2 slot=<slot_id> LoginUser=<Crypto_User_PIN> ListObjects
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./p11tool2 Slot=1 LoginUser=12345678 ListObjects

CKO_SECRET_KEY:

+ 1.1
  CKA_KEY_TYPE           = CKK_AES
  CKA_SENSITIVE          = CK_TRUE
  CKA_EXTRACTABLE        = CK_FALSE
  CKA_LABEL               = hmac_utimaco
  CKA_ID                 =
  CKA_UTC_COUNTER        = 0

+ 1.2
  CKA_KEY_TYPE           = CKK_AES
  CKA_SENSITIVE          = CK_TRUE
  CKA_EXTRACTABLE        = CK_FALSE
  CKA_LABEL               = mkek_utimaco
  CKA_ID                 =
  CKA_UTC_COUNTER        = 0
```

Figure 4 : Listing HMAC and MKEK with p11tool2

5.4 Initialize and Authorize HMAC and MKEK for CP5

A key can not be used without being authorized in CryptoServer CP5. To authorize a key, you must first initialize it with an authorization key and then set the authorization with the `AuthorizeKey` command.

1. List existing keys using `cxitool ListKeys`.

```
>_ Console

# ./cxitool Dev=3001@localhost LogonPass=<CryptoUser>,<CryptoUser_PIN>
Group=<group_name> ListKeys
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./cxitool Dev=3001@localhost LogonPass=USR_0001,12345678 Group=SL0T_0001 ListKeys
idx algo  size type  group      name      spec
-----
1  AES   256  sec   SLOT_0001
2  AES   256  sec   SLOT_0001
```

Figure 5 : Listing keys with cxitool

2. Check the status of key initialization using `cxitool KeyInfo`.

```
>_ Console

# ./cxitool dev=3001@127.0.0.1 LogonPass=<CryptoUser>,<CryptoUser_PIN>
Group=<group_name> Spec=1 KeyInfo

# ./cxitool dev=3001@127.0.0.1 LogonPass=<CryptoUser>,<CryptoUser_PIN>
Group=<group_name> Spec=2 KeyInfo
```

We can see that the key's 'Initialized' status is 'False'.

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./cxitool dev=3001@127.0.0.1 LogonPass=USR_0001,12345678 group="SL0T_0001" spec=1 Keyinfo
Group      : SL0T_0001
Name       :
Specifier  : 1
Algo       : AES
Size       : 256
Export     : 0x00000000
  Extractable : 0
  Sensitive   : 1
Usage      : 0x00000088
  Encrypt    : 0
  Decrypt    : 0
  Sign       : 0
  Verify     : 0
  Verify_Rec : 0
  Wrap       : 1
  Unwrap     : 1
  Derive     : 0
BlockLen   : 16
Type       : sec
DateGen    :
DateExp    :
Label      : mkek_utimaco
Initialized : False
```

Figure 6 : MKEK key info before initialization

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./cxitool dev=3001@127.0.0.1 LogonPass=USR_0001,12345678 group="SL0T_0001" spec=2 Keyinfo
Group      : SL0T_0001
Name       :
Specifier  : 2
Algo       : AES
Size       : 256
Export     : 0x00000000
  Extractable : 0
  Sensitive   : 1
Usage      : 0x00000022
  Encrypt    : 0
  Decrypt    : 0
  Sign       : 1
  Verify     : 1
  Verify_Rec : 0
  Wrap       : 0
  Unwrap     : 0
  Derive     : 0
BlockLen   : 16
Type       : sec
DateGen    :
DateExp    :
Label      : hmac_utimaco
Initialized : False
```

Figure 7 : HMAC key info before initialization

3. Run below command to generate user authentication key file KA.key, which contains an RSA key pair of given size. This authentication key will be used to initialize the MKEK and HMAC in the next steps.

```
>_ Console
```

```
# ./csadm GenKey=keys/KA.key,2048,"USR_0001"
```

4. Initialize the MKEK and HMAC with authentication file KA.key.

```
>_ Console

# ./cxitool Dev=3001@localhost LogonPass=<CryptoUser>,<CryptoUser_PIN>
Group=<group_name> Spec=1 Keyfile=keys/KA.key InitializeKey

# ./cxitool Dev=3001@localhost LogonPass=<CryptoUser>,<CryptoUser_PIN>
Group=<group_name> Spec=2 Keyfile=keys/KA.key InitializeKey
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./cxitool Dev=3001@localhost LogonPass=USR_0001_12345678 Group=SL0T_0001 Spec=1 Keyfile=keys/KA.key InitializeKey
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./cxitool Dev=3001@localhost LogonPass=USR_0001_12345678 Group=SL0T_0001 Spec=2 Keyfile=keys/KA.key InitializeKey
```

Figure 8 : Initialize MKEK and HMAC

5. Check the status of key initialization again using `cxitool KeyInfo`.

```
>_ Console

# ./cxitool dev=3001@127.0.0.1 LogonPass=<CryptoUser>,<CryptoUser_PIN>
Group=<group_name> Spec=1 KeyInfo

# ./cxitool dev=3001@127.0.0.1 LogonPass=<CryptoUser>,<CryptoUser_PIN>
Group=<group_name> Spec=2 KeyInfo
```

```

stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./cxtool dev=3001@127.0.0.1 LogonPass=USR_0001,12345678 group="SL0T_0001" spec=1 Keyinfo
Group      : SL0T_0001
Name       :
Specifier  : 1
Algo       : AES
Size       : 256
Export     : 0x00000000
  Extractable : 0
  Sensitive   : 1
Usage      : 0x00000088
  Encrypt    : 0
  Decrypt    : 0
  Sign       : 0
  Verify     : 0
  Verify_Rec. : 0
  Wrap       : 1
  Unwrap     : 1
  Derive     : 0
BlockLen   : 16
Type       : sec
DateGen    :
DateExp    :
Label      : mkek_utimaco
Initialized : True
Assigned   : True
Protocol   : PSS
Auth Failures : 0
Remaining Ops : 0
    
```

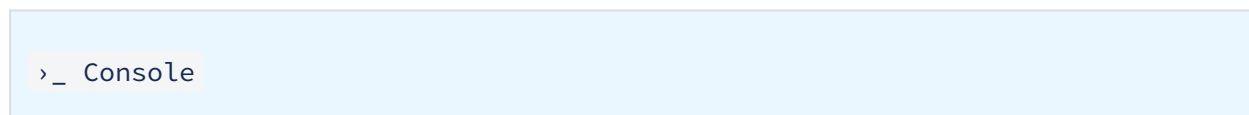
Figure 9 : MKEK key info after initialization

```

stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./cxtool dev=3001@127.0.0.1 LogonPass=USR_0001,12345678 group="SL0T_0001" spec=2 Keyinfo
Group      : SL0T_0001
Name       :
Specifier  : 2
Algo       : AES
Size       : 256
Export     : 0x00000000
  Extractable : 0
  Sensitive   : 1
Usage      : 0x00000022
  Encrypt    : 0
  Decrypt    : 0
  Sign       : 1
  Verify     : 1
  Verify_Rec. : 0
  Wrap       : 0
  Unwrap     : 0
  Derive     : 0
BlockLen   : 16
Type       : sec
DateGen    :
DateExp    :
Label      : hmac_utimaco
Initialized : True
Assigned   : True
Protocol   : PSS
Auth Failures : 0
Remaining Ops : 0
    
```

Figure 10 : HMAC key info after initialization

6. Authorize the MKEK and HMAC with authentication file KA.key.



```
# ./cxitool Dev=3001@localhost LogonPass=<CryptoUser>,<CryptoUser_PIN>  
Group=<group_name> Spec=1 Keyfile=keys/KA.key AuthorizeKey  
  
# ./cxitool Dev=3001@localhost LogonPass=<CryptoUser>,<CryptoUser_PIN>  
Group=<group_name> Spec=2 Keyfile=keys/KA.key AuthorizeKey
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./cxitool Dev=3001@localhost LogonPass=USR_0001,12345678 Group=SL0T_0001 Spec=1 Keyfile=keys/KA.key AuthorizeKey  
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./cxitool Dev=3001@localhost LogonPass=USR_0001,12345678 Group=SL0T_0001 Spec=2 Keyfile=keys/KA.key AuthorizeKey
```

Figure 11 : Authorize MKEK and HMAC



Keys need to be authorized before being used for encryption, decryption, signing, verifying and computing hashes. The command can be executed with or without `<authcounter>` parameter (e.g. `AuthorizeKey=10` or `AuthorizeKey`). The parameter defines the number of operations the cryptographic key can be used for. If the key is used for an operation, the counter is decremented by 1. When it reaches 0, it cannot be used anymore until the `AuthorizeKey` command is executed again. If we run the command without the parameter, an unlimited number of key operations is supported until a server restart or revoking the authorization.

7. Confirm the keys were authorized by using `cxitool KeyInfo` and checking the value under `Remaining Ops`.

```
>_ Console
```

```
# ./cxitool dev=3001@127.0.0.1 LogonPass=<CryptoUser>,<CryptoUser_PIN>  
Group=<group_name> Spec=1 KeyInfo  
  
# ./cxitool dev=3001@127.0.0.1 LogonPass=<CryptoUser>,<CryptoUser_PIN>  
Group=<group_name> Spec=2 KeyInfo
```

```

stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./cxitool dev=3001@127.0.0.1 LogonPass=USR_0001,12345678 group=SL0T_0001 spec=1 Keyinfo
Group      : SL0T_0001
Name       :
Specifier  : 1
Algo       : AES
Size       : 256
Export     : 0x00000000
  Extractable : 0
  Sensitive   : 1
Usage      : 0x00000088
  Encrypt     : 0
  Decrypt     : 0
  Sign        : 0
  Verify      : 0
  Verify_Rec. : 0
  Wrap        : 1
  Unwrap      : 1
  Derive      : 0
BlockLen   : 16
Type       : sec
DateGen    :
DateExp    :
Label      : mkek_utimaco
Initialized : True
Assigned   : True
Protocol   : PSS
Auth Failures : 0
Remaining Ops : Unlimited
    
```

Figure 12 : Authorized MKEK

```

stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./cxitool dev=3001@127.0.0.1 LogonPass=USR_0001,12345678 group=SL0T_0001 spec=2 Keyinfo
Group      : SL0T_0001
Name       :
Specifier  : 2
Algo       : AES
Size       : 256
Export     : 0x00000000
  Extractable : 0
  Sensitive   : 1
Usage      : 0x00000022
  Encrypt     : 0
  Decrypt     : 0
  Sign        : 1
  Verify      : 1
  Verify_Rec. : 0
  Wrap        : 0
  Unwrap      : 0
  Derive      : 0
BlockLen   : 16
Type       : sec
DateGen    :
DateExp    :
Label      : hmac_utimaco
Initialized : True
Assigned   : True
Protocol   : PSS
Auth Failures : 0
Remaining Ops : Unlimited
    
```

Figure 13 : Authorized HMAC

The keys are now initialized and authorized and can be used by OpenStack Barbican.

6 Verification and Testing

6.1 Functional Testing

6.1.1 Encrypting and Decrypting Secrets

1. Authenticate the current shell with OpenStack admin credentials.

```
>_ Console
```

```
# source ~/devstack/openrc admin admin
```

2. Create a secret or password.

```
>_ Console
```

```
# openstack secret store --name utimaco123 --payload password
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ openstack secret store --name utimaco123 --payload password
+-----+-----+
| Field | Value |
+-----+-----+
| Secret href | http://127.0.0.1/key-manager/v1/secrets/a06c9a76-704c-4c1f-998d-3956ddd700b7 |
| Name | utimaco123 |
| Created | None |
| Status | None |
| Content types | None |
| Algorithm | aes |
| Bit length | 256 |
| Secret type | opaque |
| Mode | cbc |
| Expiration | None |
+-----+-----+
```

Figure 14: Secret created

3. You can also verify the encryption operation logging in PKCS#11 log file cs_pkcs11_R2.log during secret generation as shown below.

```

stack@ub22-openstack-barbican:/opt/utimaco/bin$ tail /tmp/cs_pkcs11_R2.log
05.02.2026 16:48:43 [000005B9:000005B9] C_GetSessionInfo | T: enter...
05.02.2026 16:48:43 [000005B9:000005B9] C_GetSessionInfo | T: leave...
05.02.2026 16:48:43 [000005B9:000005B9] C_GenerateRandom | T: enter...
05.02.2026 16:48:43 [000005B9:000005B9] C_GenerateRandom | T: leave...
05.02.2026 16:48:43 [000005B9:000005B9] C_EncryptInit | T: enter...
05.02.2026 16:48:43 [000005B9:000005B9] C_EncryptInit | T: leave...
05.02.2026 16:48:43 [000005B9:000005B9] C_Encrypt | T: enter...
05.02.2026 16:48:43 [000005B9:000005B9] C_Encrypt | T: leave...
05.02.2026 16:48:43 [000005B9:000005B9] C_CloseSession | T: enter...
05.02.2026 16:48:43 [000005B9:000005B9] C_CloseSession | T: leave...

```

Figure 15 : PKCS#11 logs showing secret encryption

4. Fetch the created secret information.

>_ Console

openstack secret get <secret_href>

```

stack@ub22-openstack-barbican:/opt/utimaco/bin$ openstack secret get http://127.0.0.1/key-manager/v1/secrets/a06c9a76-704c-4c1f-998d-3956ddd700b7
+-----+-----+
| Field | Value |
+-----+-----+
| Secret href | http://127.0.0.1/key-manager/v1/secrets/a06c9a76-704c-4c1f-998d-3956ddd700b7 |
| Name | utimaco123 |
| Created | 2026-02-05T16:48:44+00:00 |
| Status | ACTIVE |
| Content types | {'default': 'application/octet-stream'} |
| Algorithm | aes |
| Bit length | 256 |
| Secret type | opaque |
| Mode | cbc |
| Expiration | None |
+-----+-----+

```

Figure 16 : Fetching secret information

5. Fetch the created secret's value.

>_ Console

openstack secret get <secret_href> --payload

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ openstack secret get http://127.0.0.1/key-manager/v1/secrets/a06c9a76-704c-4c1f-998d-3956ddd700b7 --
payload
+-----+
| Field | Value |
+-----+
| Payload | password |
+-----+
```

Figure 17 : Fetching secret's value

The secret is first decrypted and then displayed.

- You can also verify the decryption operation logging in PKCS#11 log file `cs_pkcs11_R2.log` during secret retrieval as shown below.

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ tail /tmp/cs_pkcs11_R2.log
05.02.2026 16:57:23 [000005B9:000005B9] Cxi::init | I: Cxi version 1.7.20 [Linux-x86_64] (Oct 7 2025)
05.02.2026 16:57:23 [000005B9:000005B9] C_OpenSession | T: leave...
05.02.2026 16:57:23 [000005B9:000005B9] C_GetSessionInfo | T: enter...
05.02.2026 16:57:23 [000005B9:000005B9] C_GetSessionInfo | T: leave...
05.02.2026 16:57:23 [000005B9:000005B9] C_DecryptInit | T: enter...
05.02.2026 16:57:23 [000005B9:000005B9] C_DecryptInit | T: leave...
05.02.2026 16:57:23 [000005B9:000005B9] C_Decrypt | T: enter...
05.02.2026 16:57:23 [000005B9:000005B9] C_Decrypt | T: leave...
05.02.2026 16:57:23 [000005B9:000005B9] C_CloseSession | T: enter...
05.02.2026 16:57:23 [000005B9:000005B9] C_CloseSession | T: leave...
```

Figure 18 : PKCS#11 logs showing secret decryption

6.1.2 Generating a Symmetric Key in OpenStack Barbican

- Generate a new 256-bit key using `openstack secret order create` command and store it in Barbican.

```
>_ Console

# openstack secret order create --name app_key --algorithm aes --mode ctr
--bit-length 256 --payload-content-type=application/octet-stream key
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ openstack secret order create --name app_key --algorithm aes --mode ctr --bit-length 256 --payload-c
ontent-type=application/octet-stream key
+-----+
| Field | Value |
+-----+
| Order href | http://127.0.0.1/key-manager/v1/orders/6b9b7696-6630-46a3-99d6-b134f66ef91f |
| Type | Key |
| Container href | N/A |
| Secret href | None |
| Created | None |
| Status | None |
| Error code | None |
| Error message | None |
+-----+
```

Figure 19 : Symmetric key created and stored in Barbican

- You can also verify the encryption operation logging in PKCS11 log file `cs_pkcs11_R2.log` during secret generation as shown below.

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ tail /tmp/cs_pkcs11_R2.log
05.02.2026 17:03:36 | [000005BA:000005BA] C_GenerateRandom | T: enter...
05.02.2026 17:03:36 | [000005BA:000005BA] C_GenerateRandom | T: leave...
05.02.2026 17:03:36 | [000005BA:000005BA] C_GenerateRandom | T: enter...
05.02.2026 17:03:36 | [000005BA:000005BA] C_GenerateRandom | T: leave...
05.02.2026 17:03:36 | [000005BA:000005BA] C_EncryptInit | T: enter...
05.02.2026 17:03:36 | [000005BA:000005BA] C_EncryptInit | T: leave...
05.02.2026 17:03:36 | [000005BA:000005BA] C_Encrypt | T: enter...
05.02.2026 17:03:36 | [000005BA:000005BA] C_Encrypt | T: leave...
05.02.2026 17:03:36 | [000005BA:000005BA] C_CloseSession | T: enter...
05.02.2026 17:03:36 | [000005BA:000005BA] C_CloseSession | T: leave...
```

Figure 20 : PKCS#11 logs showing key encryption

- View the details of the order to identify the location of the generated key, shown here as the `Secret href` value.

```
>_ Console

# openstack secret order get <order_href>
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ openstack secret order get http://127.0.0.1/key-manager/v1/orders/6b9b7696-6630-46a3-99d6-b134f66ef91f
+-----+-----+
| Field | Value |
+-----+-----+
| Order href | http://127.0.0.1/key-manager/v1/orders/6b9b7696-6630-46a3-99d6-b134f66ef91f |
| Type | Key |
| Container href | N/A |
| Secret href | http://127.0.0.1/key-manager/v1/secrets/e20200f5-1ee2-4dc7-858e-53a6881b8d5f |
| Created | 2026-02-05T17:03:36+00:00 |
| Status | ACTIVE |
| Error code | None |
| Error message | None |
+-----+-----+
```

Figure 21 : Viewing order details

- Retrieve the details of the secret.

```
>_ Console

# openstack secret get <secret_href>
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ openstack secret get http://127.0.0.1/key-manager/v1/secrets/e20200f5-1ee2-4dc7-858e-53a6881b8d5f
```

Field	Value
Secret href	http://127.0.0.1/key-manager/v1/secrets/e20200f5-1ee2-4dc7-858e-53a6881b8d5f
Name	app_key
Created	2026-02-05T17:03:36+00:00
Status	ACTIVE
Content types	{'default': 'application/octet-stream'}
Algorithm	aes
Bit length	256
Secret type	symmetric
Mode	ctr
Expiration	None

Figure 22 : Viewing secret details

- Alternatively, you can list the symmetric key that has been generated by the command below.

```
>_ Console

# openstack secret list
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ openstack secret list
```

Secret href	Name	Created	Status	Content types	Algorithm	Bit length	Secret type	Mode	Expiration
http://127.0.0.1/key-manager/v1/secrets/a06c9a76-704c-4c1f-998d-3956ddd700b7	utimaco123	2026-02-05T16:48:44+00:00	ACTIVE	{'default': 'application/octet-stream'}	aes	256	opaque	cbc	None
http://127.0.0.1/key-manager/v1/secrets/e20200f5-1ee2-4dc7-858e-53a6881b8d5f	app_key	2026-02-05T17:03:36+00:00	ACTIVE	{'default': 'application/octet-stream'}	aes	256	symmetric	ctr	None

Figure 23 : Symmetric key in secret list

6.1.3 Storing Public Key, Private Key and Certificate in OpenStack Barbican

- Create a self-signed certificate using the command below.

```
>_ Console
```

```
# openssl req -x509 -newkey rsa:4096 -keyout private.pem -out cert.pem  
-sha256 -days 365 -nodes
```



The key and certificate can be generated using other utilities as well.

2. Verify the private key and certificate file are generated.

>_ Console

```
# ll
```

```
stack@ub22-openstack-barbican:~/ssl$ ll  
total 16  
drwxrwxr-x 2 stack stack 4096 Feb  5 17:38 ./  
drwxr-x--- 8 stack stack 4096 Feb  5 17:02 ../  
-rw-rw-r-- 1 stack stack 2098 Feb  5 17:38 cert.pem  
-rw----- 1 stack stack 3272 Feb  5 17:29 private.pem
```

Figure 24 : Certificate and private key created

3. Generate the public key from private key.

>_ Console

```
# openssl rsa -in private.pem -pubout -out public.pem
```

```
stack@ub22-openstack-barbican:~/ssl$ openssl rsa -in private.pem -pubout -out public.pem  
writing RSA key  
stack@ub22-openstack-barbican:~/ssl$ ll  
total 20  
drwxrwxr-x 2 stack stack 4096 Feb  5 17:43 ./  
drwxr-x--- 8 stack stack 4096 Feb  5 17:02 ../  
-rw-rw-r-- 1 stack stack 2098 Feb  5 17:38 cert.pem  
-rw----- 1 stack stack 3272 Feb  5 17:29 private.pem  
-rw-rw-r-- 1 stack stack  800 Feb  5 17:43 public.pem
```

Figure 25 : Public key created form private key

4. Store the public key in OpenStack Barbican.

```
>_ Console

# openstack secret store --algorithm rsa --secret-type public --payload-
content-type application/octet-stream --payload-content-encoding base64 --
payload "$(base64 < public.pem)" --bit-length 2048 --name pubtest
```

```
stack@ub22-openstack-barbican:~/ssl$ openstack secret store --algorithm rsa --secret-type public --payload-content-type application/octet-stream --p
ayload-content-encoding base64 --payload "$(base64 < public.pem)" --bit-length 2048 --name pubtest
+-----+-----+
| Field | Value |
+-----+-----+
| Secret href | http://127.0.0.1/key-manager/v1/secrets/4c7903ac-3984-4ba7-a19d-93b3083beff8 |
| Name | pubtest |
| Created | None |
| Status | None |
| Content types | {'default': 'application/octet-stream'} |
| Algorithm | rsa |
| Bit length | 2048 |
| Secret type | public |
| Mode | cbc |
| Expiration | None |
+-----+-----+
```

Figure 26 : Public key stored in Barbican

5. You can also verify the encryption operation logging in PKCS#11 log file `cs_pkcs11_R2.log` during public secret generation as shown below.

```
stack@ub22-openstack-barbican:~/ssl$ tail /tmp/cs_pkcs11_R2.log
05.02.2026 17:47:06 [000005BA:000005BA] C_GetSessionInfo T: enter...
05.02.2026 17:47:06 [000005BA:000005BA] C_GetSessionInfo T: leave...
05.02.2026 17:47:06 [000005BA:000005BA] C_GenerateRandom T: enter...
05.02.2026 17:47:06 [000005BA:000005BA] C_GenerateRandom T: leave...
05.02.2026 17:47:06 [000005BA:000005BA] C_EncryptInit T: enter...
05.02.2026 17:47:06 [000005BA:000005BA] C_EncryptInit T: leave...
05.02.2026 17:47:06 [000005BA:000005BA] C_Encrypt T: enter...
05.02.2026 17:47:06 [000005BA:000005BA] C_Encrypt T: leave...
05.02.2026 17:47:06 [000005BA:000005BA] C_CloseSession T: enter...
05.02.2026 17:47:06 [000005BA:000005BA] C_CloseSession T: leave...
```

Figure 27 : PKCS#11 logs showing secret encryption

6. Get value of the public key.

```
>_ Console

# openstack secret get -p -c Payload -f value <secret_href>
```

```
stack@ub22-openstack-barbican:~/ssl$ openstack secret get -p -c Payload -f value http://127.0.0.1/key-manager/v1/secrets/4c7903ac-3984-4ba7-a19d-93b3083beff8
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAACAg8AMIICCGKCAgEAQsRvnodQeXd5e9RzhvCJ
w1ZKpsT xm0H88v8Tv+hrJhzZC88BoheYaoz+S1QFVMqGHuouPdNrzqFU1Un7Mwt9
BHg30tMnHCqjuAvml04ujGjhSSS4IFF+W/fR2dF5ncp/X2pA0kgyxL3dNIRCWZjy
ZFUuNRdIyGSHS4xnDagPUje8YegcV6+VVB8YHA6m8gT1LjRhoSSqZig7+0zVxaZk
VCWzzt8UGdggoktmVhyU42AoC2+L0TFP13cxPEwLDF8qzJe+YV93w0FNvYVPBfZW
JQ0T591ydZu0a06vsXhu4275ZLSYJw+jzjFNi/qygo9A28SUYur2GS1Zv8xjtlk1
s+MqymPq//SouHbkIUAhgds0+zCV3nXmcFMqtbWJ3qaLN+mk+9ZB8xH04K8t/r1P
q56TTrdIuTHGMvj0DrT094wvqksztwqtB67TFe00zFORx5DFyUVUYUTVL0JjQzUt
baPEF4ustxdKUfSkf6n04ah1UD8DH12zGkzVMhtwMTtnSLYvnF6Hoy8EgI50NkZa
aAJG/01sV6jN/y00e98xficds5bwymlYrgAtsydd1z/T4h035dpj/I0hrttmoE6s
FU9M4o5sda/c6v20SYk19pNLKKAxkFufxrXc1aDaF0eejE29Hth1An10Zsuyt
w4r6EFu7ueX28cJ1062R5PECAwEAQAQ=
-----END PUBLIC KEY-----
```

Figure 28 : Get value of the public key from Barbican

7. Store private key in OpenStack Barbican.

```
>_ Console

# openstack secret store --algorithm rsa --secret-type private --payload-
content-type application/octet-stream --payload-content-encoding base64 --
payload "$(base64 < private.pem)" --bit-length 2048 --name privatekeytest
```

```
stack@ub22-openstack-barbican:~/ssl$ openstack secret store --algorithm rsa --secret-type private --payload-content-type application/octet-stream --
payload-content-encoding base64 --payload "$(base64 < private.pem)" --bit-length 2048 --name privatekeytest
```

Field	Value
Secret href	http://127.0.0.1/key-manager/v1/secrets/92bc70b9-4574-41ba-9973-508e4fbd5c37
Name	privatekeytest
Created	None
Status	None
Content types	{'default': 'application/octet-stream'}
Algorithm	rsa
Bit length	2048
Secret type	private
Mode	cbc
Expiration	None

Figure 29 : Private key stored in Barbican

8. Get value of the private key.

```

_ _ Console

# openstack secret get -p -c Payload -f value <secret_href>

stack@ub22-openstack-barbican:~/ssl$ openstack secret get -p -c Payload -f value http://127.0.0.1/key-manager/v1/secrets/92bc70b9-4574-41ba-9973-508e4fbd5c37
----BEGIN PRIVATE KEY-----
MIIEJRAIBADANBgkqhkiG9w0BAQEFAASCC4wggkqAgEAAoICAQCpKu+eh1B5d3L7
1H0G8InDvmSmxPgbQfzy/x0/6GsmHnkLzWg1F5hjqP5LVAUyoYe6i4902V0oVSJ
SfszC30Eedc60yccKq04C+Ys7i6MYmFJJLggUX5b99HZ0VKcKn9fakDSSDLGX00
HEJZmpJKw41F0jIZIdKHjGcNqo9SN7xh6BXR5VUHxgcDqbyBPUUNgGhJKpk1Dv4
7NXFpmS8JbP03xQZ2CC1S2ZWHJTYCglb4s5MU+XdzE8TAsHXyRmL75hX3FD0U29
hu8F/ZYLDRPn3XJ1m7R07q+xeG7jvLktJgnD6NmMU2L+rKcj0DbxJTK6vYZKJm/
zG0ZTWz4x0KY+r/9I64ds0hQCEZ2w77MJXedeZUyqJtYnepos36Yr71KHzEftg
ry3+vu+pLn0t0i5McYy+NA0u3T3ha+q5z03Cq0HrTMV447MU5SHKJ/RJVRhRNUV
QmDNS1to8QX16y3F2RR/mR/qc7hoeVQpWfXbMaT08Y63AX02dItt+cXoeJLw5A
hlQ2RlpoAkB/TWY/qM5/LQ573zF+z2ZLV0KYthtCoC2zJ13XP9P1E7fL2kn8g6FG
2zagTox0T0zh2jm1+r9zq/ah3j1TX2k2U0oBfGQMS/GtdyJ0NoXR56Mfb0eKtGcl
R1KTLFPDivoR+7u55fpxmmlTfZihk80IDAQBAoICAAP1C2KbnXMS5AT98p0VL1
Gus9LzKXpkhf2SwcXedqV/P7ea6oRoAhkzXmneFSSTh80i3dVA75Jt0gy43n5ua
BpQ1shP7M/qsmELpMjV2gv0R437HkuVyc63m0Puy5WeSRXwXorkRpuDm88q0KMe
Wfnt+8bf2UMmN6nptPSGeeXge0rJ0MxGICL84gsLEDMIJ93iXp+0w8ncYVhf8yj8
+uaT5EjnyuITJkwjth/FtVn0pyvpxxiUiG+Eel_9hzQ+NSThIxpMeF5u36wmm0d
0+q13GkhHg3xcE43gq1+bxSDvrmAUCw/ZrZUV0pKN0W6J0gGk973c4U1031Dtps
M289Xr4DxwRL18Y5zqJWfe03NyrByoI5NSP/J38UYK0X4J9f90rZ6P7fz5n7
XzHaw4HsnFoVValiJcTY2c+ke55bq8gNPHk4pzTNDquS6Pvbcv/f6gqN+thFaate
LSgrBTnpgKQFwycovLzFdN1S23LRJCQ1kZB5qqPurjTnELg215oKvMYtoz6berb
x0R9i0jX603sG0fu+3kmBWLx0dcvtvIm0pRr3e+2cvZhv6wrptd089/ELt1Q05G2
j0t2e2ZE0AS9b6ulhbJ7ZpgMaH/K0NFLR41NFxSbXJirb3PcVBH9PvSRJGfAYI
E+H/wpATGx30dUoSN4Zha0TBAQ0iDC4wog0+kLQUC3wXAS7doEjiVA0S4CP2cwzZ
CzeJ0gmV60jjfSbWYIX0XGJ08E8nCmXkks+2KEJw3d/RFYAI/5k3jpp8DLWlUkh
ksZl2bw17Rvr+u0WAV5ftar0Vp0CDCLPHVg7T58ETFH52L/PJurLimJOV1H0JW0
Z2XwiChLiy4LN0awDyYktItS9Ge030BaTo/Bj51exjI8S6bKHCCwRLYfL0Udyhw
kFbJGK0ntxS6L2uyV9mpanluctABQpp05MGI0M5GekR1G9LN2nAv2AM3/LEYHuW
fe6Pg81W+TcPolGkVr0kXNU0xiKArCPDMyr6xst/+WB+4HRAoIBAQC/LV60Buzf
QF5mT2K041JCSq34y4iipfYn/IMNY8QVW0e000n0v6KEGrfaprgoTq33u/nzXl
92V49RYTak0gk1gkwpDPVbDobaMSP2CCBf0XHHSKJ4ybhI050twJYw6653yv3d2
T0YSuywrlLPEeINLbkkZ62wnSGDdYqj9RVMSWyhXHy5v3TJ8DnsPP1ChxZ1K0P0
peDPolXDxb9XHGEjYpVlHqkeZ7A6m8i0kfnuW5GjXLe9Ns03RYjd2HpGkDAfak3
cy2A7rT1pMUFhjILsgctb0T4K8mvzNcRHTGtujaJ5assOnLNsIzzyeAicMiVvef
4TK+YwzUXNkhAoIBA0DY7K9d80T5c10xNsRcn1Q0Q0cu50mSrZ5VZ1sw/L7KDzKL
uFseVgjL24MxvjI54Kc0LviKpH/JM8J2yCno6/9L6z66NbeQaH2gmbjCK0RRQXez
fsmI0R0E2GnFad9Ca4mp0Jo+N6K8/W81SjgHRbHVf06dP2Fk2rP1iLAu/8sH5bHA
+7rW96Sng0ujb2JbLaNHEKTFBEJrk0Znf1Xtn+YF5+0JwRU9P1jbxu3jBGKEwrXw
ovU1J0TGxSH2bMcqMuwCCRZjmYtpYQWsynsK50rGN5Xpk/aix9B7QUv63YgLVm2
7xTdv3tZFNrehmfkWTYH/LkdAeArIiz0yLAdkfbAoIBAQCOnMeaHIGdwpTozDVs
8wIAAqvjA0Dp44NPRRj4D7J0dsvXrdSGC9Pz8z0jNu00na1lgPX59v1WPwXQ1bH
QZ0yM7RQkJd0070B+wMmMQNgRblkbrx7EG1amuckYvab/8dYXw0+n9p8nLxK0ZyR
IbEky/zID2BQ6dZwp4gUgYnT/QomMps7KQyYnTVALqH1GlaRR+Pp5km1boY
0SP144RHw+3gr9gagxe3JYce2HoP7sqz2D2HTHW06PK+6fjfhCRpzeFeHoXj8U2R
gdsL1q5jZagF7RkS+XJIRFjgfs5nadVIRqjrmqLshelLq1UeLhSpzATfhpZaV2
m7shAoIBA0D0j8GyTZ9vjy83TDfBJLycQChf3KT0AXF8unMA+Xf8zI5o8C1F+4eS
2MH3qwZ0jK6d0TVH2H7MIdraT+Sx3U0gsrCYnUUHQbdS+aeblLMadmu19t6AA1L
oVqy+VqbUUXMm5j0Gb8s25G03PWYXKbovyKoAjcnI5j6jXbfl7sh1Y5ypYMSJ2ge
3Vt0/t+MKSVS2/C2GzU7prZ0u09FrcIT0a/a1fFB5nmJ2dxrbg5nUL/1b0aCQLAE
sTTU40ablQKc5i3xy+kjwIAFP/G2togwFU4YdkK05TEicHcKagUEHfj2d945Gke
7s2/hej9teauXrRb04F0b1Z0Mpt7BJCF
----END PRIVATE KEY-----

```

Figure 30 : Get value of the public key from Barbican

9. Store the certificate in OpenStack Barbican.

```

_ _ Console

```

```
# openstack secret store --algorithm rsa --secret-type certificate --
payload-content-type application/octet-stream --payload-content-encoding
base64 --payload "$(base64 < cert.pem)" --bit-length 2048 --name
utimacocertificatetest
```

```
stack@ub22-openstack-barbican:~/ssl$ openstack secret store --algorithm rsa --secret-type certificate --payload-content-type application/octet-stream --payload-content-encoding base64 --payload "$(base64 < cert.pem)" --bit-length 2048 --name utimacocertificatetest
```

Field	Value
Secret href	http://127.0.0.1/key-manager/v1/secrets/310f6e24-2f4b-404b-952c-0c08042e5f78
Name	utimacocertificatetest
Created	None
Status	None
Content types	{'default': 'application/octet-stream'}
Algorithm	rsa
Bit length	2048
Secret type	certificate
Mode	cbc
Expiration	None

Figure 31 : Certificate stored in Barbican

10. Get value of the certificate from OpenStack Barbican.

```
> _ Console
```

```
# openstack secret get -p -c Payload -f value <secret_href>
```

```
stack@ub22-opensack-barbican:~/ssl$ openssl secret get -p -c Payload -f value http://127.0.0.1/key-manager/v1/secrets/310f6e24-2f4b-404b-952c-0c08042e5f78
-----BEGIN CERTIFICATE-----
MIIF3zCCABegAwIBAgIUMLFS03FD9B6IxxQ/Koht4XiSL90wDQYJKoZIhvcNAQEL
BQAwfzELMAkGA1UEBHMCREUxHDAaBgNVBAGME05vcmRyaGVpb2V1X2ZmFzZW4x
DzANBgNVBAMkMBkFhY2h1b2JlQWMA4GA1UECgwHVXRpbWVjby5jb20wHhcNMjYw
MjY1MTczODM4wDQYJKoZIhvcNAQEBBQADGAPADCCAgQcggIBAKKq756HUH13eXVU
c4bwiNWZkBE8ZtB/PL/E7/oayYc2QvPAAIXmGqM/ktUBTKhh7qLj3Ta86hVIJ
+2MLfQR4NzrTJxwqo7gLSizuloxiYUkkuCBFRfLv30dnRUpwqf19qQNJIMSz3TSE
QLmY8mRbJUXSMhkh0eMzW2qj1I3vGhHfVlVQfGBwOpvIE9540YaEkqmsIO/jS
1cWmZLwLS87fFBnYIKJLZLYcLONgKAtvzkXT5d3MTxMCwxFksyXvmFfd8NBtB2F
TWX9LlUNE+fdcnwbtGjur7F4buNu+W50mccPozYxTYv6soKPNvELMrq9hKomb/M
Y7ZZNBpjEpmj6v/0jrh2yiFAIRnb0vswld515nBTKom1id6mizfpivvWQfMR90cv
LF69T6kuk063SLkxjL40A67dPeFr6pLM7cKrq0eu0xxjjsxTKceQ38LFVGF61S9C
Y0M1LW2jxBeLrLcXZFh+ZHpZuGh5VA/Ax9dsxpM7zIbcDE7Z0i2L5xeh0MvBICE
EDZGwmgCRv9Nbl+ozf8tDnvfMx4nHb0w8Mp12GkgLbMnXdc/8+IT+Xa5fyDoUbb
ZqB0RHP1TOHa0bHwv30r9qEmJNfaTZ5Igf8ZBbn8a13Img2hdHnoxNVR40yG4T6
UrssU80k+hh7u7n18/HcytoktetXagMBAAGjUzBRMB0GA1UdDgQWBQaQdJUCeG2
Q0ICzXLSFwUAgeJVHjAfbgNVHSMGDAwG80aQdJUCeG2Q0ICzXLSFwUAgeJVHjAP
BgnVHRMBA8EBTADAQH/MA0GCSqGSIb3DQEBCwUAA4ICAQBYkEzr5NVFsIE+j/k
+r0uBE4B01GE6E90DKjNf5hM0EfmXaRHS1/Htz0DF5f+gzRjDAA30yIPa+BgtK
R5sy9mMnUJae0e5d7v1hxs0glnuZFNmHf68vx07sqYfG4M1408zaUpzt/NrFhQx
IDAg1Mh4eKtR89VCWATS/efmStXt+HSD8zFa0NZL186nNaxMkhWj10Epa7T7V
K7iSklejUYp39w3i373BSWjQKP/P+huzj3ZB8KjN7H04CC0ACfct0a6uipMyG/8
aY01IokosipkTwRpyfF30FnfoSjg/b11yurFrHVt2VJU5XZjellLsej+I6D1TsVBR
cba0d2t3kShjrFdzJd1F9NCBB8oxWdCLdbwz2yKssuHeRY4CTKJoJN61S0Vm2
xaIPgEe3cGHFXJ05s6TVerHwJVerh5XT0540YSyITHV0AwJ70gve1Xj1oyMy9a
a0B23XAY1NLcYVaizFDjfyty2Tca1VkoakHqD4SjUPNa01N0Hb6p0Mkww5U63RG
M0M1ndJCQa5UY9Xik31iYm93avd0Lwsg8WpMA641w13Z1s6BPwtckIijywxvdV4
y5y5i66bThXfHwq8gg9wfo3q0vqf/y18YH3Y1kHUKdZQ20JmW+HW15b6M0LVR0
v+Be+ExpS+9p20AHTXbz0sZyG==
-----END CERTIFICATE-----
```

Figure 32 : Get value of certificate from Barbican

11. List all secrets.

```
>_ Console

# openssl secret list
```

```
stack@ub22-opensack-barbican:~/ssl$ openssl secret list
```

Secret href	Name	Created	Status	Content types	Algorithm	Bit length	Secret type	Mode	Expiration
http://127.0.0.1/key-manager/v1/secrets/310f6e24-2f4b-404b-952c-0c08042e5f78	utimacocertificatetest	2026-02-06T07:53:43+00:00	ACTIVE	{'default': 'application/octet-stream'}	rsa	2048	certificate	cbc	None
http://127.0.0.1/key-manager/v1/secrets/4c7903ac-3984-4ba7-a19d-93b3083beff8	pubtest	2026-02-05T17:47:06+00:00	ACTIVE	{'default': 'application/octet-stream'}	rsa	2048	public	cbc	None
http://127.0.0.1/key-manager/v1/secrets/92bc70b9-4574-41ba-9973-508e4fd5c37	privatekeytest	2026-02-06T07:47:09+00:00	ACTIVE	{'default': 'application/octet-stream'}	rsa	2048	private	cbc	None
http://127.0.0.1/key-manager/v1/secrets/a06c9a76-704c-4c1f-998d-3956ddd700b7	utimaco123	2026-02-05T16:48:44+00:00	ACTIVE	{'default': 'application/octet-stream'}	aes	256	opaque	cbc	None
http://127.0.0.1/key-manager/v1/secrets/420200f5-1ee2-4dc7-850e-53a6881b8d5f	app_key	2026-02-05T17:03:36+00:00	ACTIVE	{'default': 'application/octet-stream'}	aes	256	symmetric	ctr	None

Figure 33 : Listing all secrets

6.1.4 Key Rotation/Migration



KEK Rewrap Compatibility

In OpenStack Barbican Dalmatian, the PKCS#11 `unwrap_key()` interface requires an explicit encryption mechanism parameter, while the `rewrap_pkek` utility still uses an older call signature. Without alignment, KEK rewrapping fails. A small compatibility fix is required to pass the configured encryption mechanism to `unwrap_key()`.

The compatibility fix is performed with the following steps:

1. Locate the `pkcs11_kek_rewrap.py` script (default location is `/opt/stack/barbican/barbican/cmd/pkcs11_kek_rewrap.py`).
2. Find the original code.

```
pkcs11_kek_rewrap.py
```

```
current_kek = self.pkcs11.unwrap_key(kek_mkek, iv, wrapped_key, session)
```

3. Update the code.

```
pkcs11_kek_rewrap.py
```

```
current_kek = self.pkcs11.unwrap_key(self.pkcs11.key_wrap_mechanism,  
kek_mkek, iv, wrapped_key, session)
```

4. Save the changes.

Perform the key rotation using the following steps:

1. Generate a new MKEK using the `p11tool2 GenerateKey` command below.

```
>_ Console

# ./p11tool2 Slot=<slot_id> LoginUser=<CryptoUser_PIN>
KeyAttr=CKA_LABEL="mkek_utimaco1234",CKA_VALUE_LEN=32,CKA_WRAP=true,CKA_UNWRAP=true,CKA_ENCRYPT=false,CKA_DECRYPT=false,CKA_EXTRACTABLE=false
GenerateKey=AES
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./p11tool2 Slot=1 LoginUser=12345678 KeyAttr=CKA_LABEL="mkek_utimaco1234",CKA_VALUE_LEN=32,CKA_WRAP=true,CKA_UNWRAP=true,CKA_ENCRYPT=false,CKA_DECRYPT=false,CKA_EXTRACTABLE=false GenerateKey=AES
```

Figure 34 : New MKEK generated

2. Generate a new HMAC using the `barbican-manage hsm gen_hmac` command.

```
>_ Console

# sudo -u stack -E barbican-manage hsm gen_hmac --library-path '/opt/utimaco/lib/libcs_pkcs11_R2.so' --passphrase <CryptoUser_PIN> --slot-id <slot_id> --label 'hmac_utimaco1234' --length 32
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ sudo -u stack -E barbican-manage hsm gen_hmac --library-path '/opt/utimaco/lib/libcs_pkcs11_R2.so' --passphrase 12345678 --slot-id 1 --label 'hmac_utimaco1234' --length 32
HMAC successfully generated!
```

Figure 35 : New HMAC generated

3. Verify that the keys are generated on the Utimaco HSM using the `p11tool2 ListObjects` command.

```
>_ Console

# ./p11tool2 slot=<slot_id> LoginUser=<CryptoUser_PIN> ListObjects
```

```
+ 1.5
CKA_KEY_TYPE           = CKK_AES
CKA_SENSITIVE          = CK_TRUE
CKA_EXTRACTABLE        = CK_FALSE
CKA_LABEL              = mkek_utimaco1234
CKA_ID                 =
CKA_UTILITY_COUNTER    = 0
```

Figure 36 : Listing keys with p11tool2 (MKEK)

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ ./p11tool2 slot=1 LoginUser=12345678 ListObjects

CKO_SECRET_KEY:

+ 1.1
CKA_KEY_TYPE           = CKK_AES
CKA_SENSITIVE          = CK_TRUE
CKA_EXTRACTABLE        = CK_TRUE
CKA_LABEL              =
CKA_ID                 =
CKA_UTILITY_COUNTER    = 0

+ 1.2
CKA_KEY_TYPE           = CKK_AES
CKA_SENSITIVE          = CK_TRUE
CKA_EXTRACTABLE        = CK_TRUE
CKA_LABEL              =
CKA_ID                 =
CKA_UTILITY_COUNTER    = 0

+ 1.3
CKA_KEY_TYPE           = CKK_AES
CKA_SENSITIVE          = CK_TRUE
CKA_EXTRACTABLE        = CK_FALSE
CKA_LABEL              = hmac_utimaco1234 ←
CKA_ID                 =
CKA_UTILITY_COUNTER    = 0
```

Figure 37 : Listing keys with p11tool2 (HMAC)

4. Initialize and authorize the new keys using the procedure described in [Initialize and Authorize HMAC and MKEK for CP5](#). The same authorization key `KA.key` can be used.
5. Open `barbican.conf` and change the 'mkek_label' and 'hmac_label' values to the labels of keys created in previous steps.

```
barbican.conf
```

```
# Master Key Encryption Key and HMAC labels
mkek_label = mkek_utimaco1234
mkek_length = 32
hmac_label = hmac_utimaco1234
```

6. Restart OpenStack Barbican services.

```
> _ Console

# sudo systemctl restart devstack@barbican*
```

7. Run `rewrap_pkek` command to rewrap PKEK with the new MKEK.

```
> _ Console

# sudo -u stack -E barbican-manage hsm rewrap_pkek
```

```
2026-02-06 10:10:31.909 INFO dbcounter [-] Registered counter for database barbican
Retrieving all available projects
Retrieving KEKs for Project bb98c4ee-0b1d-4cf4-8cf9-667b08bea4f0
Post-change IV: b'dPsM5BDwVhSzfrQaks+TvW==', Wrapped Key: b'tJYRpobs8ZTD3bRzabeslPFCzmVAoPEYTZa3I/D+AjAu0XKLRlg3i8ssbAVuk5PY'
```

Figure 38 : Rewrapping PKEK

8. Verify that you can get value of the secret generated earlier.

```
> _ Console

# openstack secret get <secret_href> --payload
```

```
stack@ub22-openstack-barbican:/opt/utimaco/bin$ openstack secret get http://127.0.0.1/key-manager/v1/secrets/a06c9a76-704c-4c1f-998d-3956ddd700b7 --
payload
+-----+
| Field | Value |
+-----+
| Payload | password |
+-----+
```

Figure 39 : Fetching secret's value after PKEK rewrap

7 Troubleshooting

7.1 Common Issues and How To Resolve Them

Error	Diagnosis
<p>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)</p>	<p>PKCS#11 Slot is not initialized.</p>
<p>Error:Slot 0000 0000: p11cat.P11.getAuthState(Native Method): CS_GetSessionInfo returned Error 0x00000030 (CKR_DEVICE_ERROR)</p>	<ol style="list-style-type: none"> 1. Verify HSM services are up and running. 2. Check the IP entry is correct in <code>cs_pkcs11_R2.cfg</code> file.
<p>Internal Server Error: Secret payload retrieval failure seen - please contact site administrator.</p>	<ol style="list-style-type: none"> 1. Check <code>barbican.log</code> if HMAC and MKEK labels are correct. 2. Check that HMAC and MKEK are initialized and authorized using <code>cxitool KeyInfo</code>
<p>Missing value auth-url required for auth plugin password</p>	<p>OpenStack admin credentials are not loaded. Resolve by running <code># source ~/devstack/openrc admin admin</code></p>

Table 6: List of Errors and their Diagnoses

7.2 Log Location and Interpretation

7.2.1 PKCS#11 Logs

Enabling PKCS#11 logging to facilitate easier testing and troubleshooting is recommended. This can be done by configuring the `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.
- `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_R2.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 `LogLevel` to `1` or `2` to limit logging to only critical or important messages, thereby optimizing performance and reducing unnecessary log data.

7.2.2 OpenStack Barbican Logs

To facilitate testing and troubleshooting during integration, it is recommended to enable verbose logging in Barbican. Logging behavior is controlled through the `barbican.conf` configuration file. During initial setup and debugging, the global log level can be set to `debug = true` to provide detailed insight into request handling, PKCS#11 interactions, and cryptographic operations. Barbican logs are written to the DevStack log directory (typically `/opt/stack/logs/`), with the primary log file named `barbican.log`.

The following configuration options are commonly used:

- `debug = true`
Enables debug-level logging.
- `use_syslog = false`
Ensures logs are written to file rather than the system journal.

Once testing is complete, it is advisable to set `debug = false` to limit log volume and reduce operational overhead.

8 Further Information

This document forms a part of the information and support that is provided by Utimaco IS GmbH. Additional documentation can be found on the product CD in the Documentation directory.

All SecurityServer product documentation is also available at the Utimaco IS GmbH website:

<https://utimaco.com/>

8.1 References

Reference	Title/Company
[CSADMIN]	ustrust_Anchor_cHSM_Manual_Administrators.pdf

Table 7: References