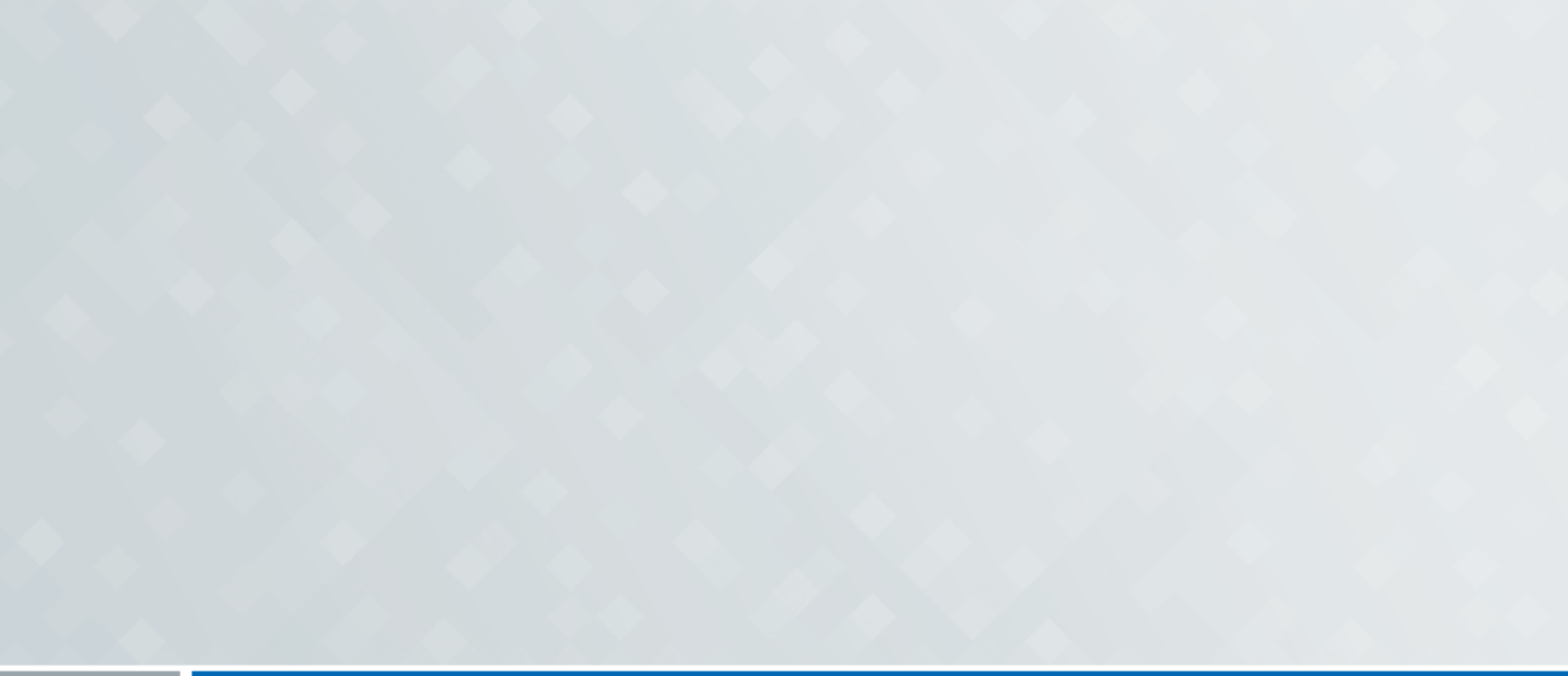




Microsoft

Azure Key Vault BYOK

2.87.0

Integration Guide

u.trust GP HSM

6.5.0

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	2026-07-01
Status	PUBLISHED
Document No.	IG-2026-0062
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	5
1.5	Document Conventions	7
2	Product Overview	9
2.1	Azure Key Vault	9
2.2	Utimaco u.trust GP HSM	9
2.3	Joint Value Proposition	9
3	Integration Requirements and Prerequisites	10
3.1	Tested Versions	10
3.2	Software Requirements	10
3.3	Hardware Requirements	11
3.4	Prerequisites	11
4	Installing and Configuring Utimaco SecurityServer Software	13
4.1	On Linux	13
4.1.1	Download and Install Utimaco Software	13
4.1.2	u.trust GP HSM PKCS#11 Configuration	14
4.1.3	Create SO, User and Initialize a Slot	16
4.2	On Windows	16
4.2.1	Update cs_pkcs11_R3.cfg	16
5	Implementing BYOK	18
5.1	Azure CLI Installation	18
5.1.1	Azure CLI Installation on Linux	18
5.1.2	Azure CLI Installation on Windows	20
5.2	Azure Key Vault Premium Subscription	21
5.3	Download Utimaco byoktool	21
5.4	Creating a Key Vault in Azure	22
5.4.1	Creating Azure Key Vault with Azure Portal	22
5.4.2	Creating Azure Key Vault on Linux	26

5.4.3	Creating Azure Key Vault on Windows	28
5.5	Generating Key Exchange Key (KEK).....	30
5.5.1	Creating a KEK with Azure Portal	31
5.5.2	Creating a KEK on Linux	35
5.5.3	Creating a KEK on Windows	36
5.6	Downloading KEK Public Key	37
5.6.1	Downloading KEK Public Key with Azure Portal	37
5.6.2	Downloading KEK Public Key on Linux	38
5.6.3	Downloading KEK Public Key on Windows	38
5.7	Generating and Preparing your Tenant Key.....	39
5.7.1	Generating Tenant Key on Linux	39
5.7.2	Generating Tenant Key on Windows.....	42
5.8	Importing Tenant Key to Azure Key Vault.....	45
5.8.1	Importing on Linux	45
5.8.2	Importing on Windows	49
5.9	FIPS mode	53
6	Troubleshooting	54
7	Further Information	55
8	Contact and Support Information	56
9	Appendices	57
9.1	References	57
9.2	Command Summary	57

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 provides instructions for integrating Utimaco u.trust General Purpose Hardware Security Module (GP HSM) with Microsoft Azure Key Vault using the Bring Your Own Key (BYOK) approach. It describes the process of securely generating and transferring cryptographic keys from the Utimaco HSM into Azure Key Vault while ensuring compliance with security and key management best practices.

1.2 Target Audience

This guide is intended for Azure administrators and HSM administrators.

1.3 Purpose of the Integration

The purpose of this integration is to enable secure generation and management of cryptographic keys within the Utimaco u.trust GP HSM, while allowing their controlled usage in Azure Key Vault through the Bring Your Own Key (BYOK) model, ensuring enhanced security, compliance, and ownership of key material.

1.4 Abbreviations

The following abbreviations are used in this guide:

Abbreviation	Meaning
API	Application Programming Interface

Abbreviation	Meaning
BYOK	Bring Your Own Key
CD	Compact Disc
CLI	Command line interface
CNG	Cryptography API Next Generation
CXI	Cryptographic eXtended Services Interface
GUI	Graphical User Interface
HSM	Hardware Security Module
JCE	Java Cryptography Extension
KEK	Key Exchange Key
LAN	Local Area Network
MBK	Master Backup Key
PCIe	PCI Express Interface
PKCS#11	Public-Key Cryptography Standard #11
RSA	Rivest-Shamir-Adleman
SKU	Stock-Keeping-Unit

Abbreviation	Meaning
ECC	Elliptic Curve Cryptography
RHEL	Red Hat Enterprise Linux
NIST	National Institute of Standards and Technology
CentOS	Community ENTERprise Operating System

Table 1: List of 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
<code>Monospaced</code>	Code that is given for explanation or as an example, file paths	You will find the file <code>example.conf</code> in the <code>/exmp/demo/</code> directory.
<i>Italic</i>	References and important terms	https://utimaco.com/

Table 2: Document conventions

Special icons are used 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 marks the result expected after the successful execution of an instruction.

2 Product Overview

2.1 Azure Key Vault

Azure Key Vault is a cloud service for securely storing and accessing "secrets". A secret is any information that you want to carefully control access to. Such examples can be API keys, passwords, certificates, or cryptographic keys. The Key Vault service supports two types of containers: vaults and managed HSM pools. Vaults support storing software and HSM backed keys, secrets, and certificates. Managed HSM pools only support HSM-backed keys which need an HSM. See [Azure Key Vault REST API](#) overview for complete details.

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

This integration combines the secure key management capabilities of Utimaco u.trust GP HSM with the scalability and flexibility of Azure Key Vault, enabling customers to maintain full control over their cryptographic keys while leveraging cloud-based services for application security and data protection.

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 you already have a working Azure subscription and users have been created on the portal to be able to configure the Azure Key Vault.

3.1 Tested Versions

The Azure Key Vault BYOK integrations that have been successfully tested with the Utimaco HSM.

Operating System	Software Requirements	Utimaco Security Server Version	Utimaco HSM
Windows 10 Enterprise Rocky Linux 9.6	BYOK tool 1.0.0	SecurityServer V6.5.0	u.trust GP HSM CSe-Series/Se-Series

Table 3: List of tested versions



This integration is not supported with external keystore.

3.2 Software Requirements

Software	Software Requirements
Operating system	Windows 64 bit, Linux 64bit
BYOK tool	byoktool 1.0.0 developed by Utimaco
Java	Version 8, Update 271 or higher

Software	Software Requirements
P11tool2	PKCS#11 administration tool developed by Utimaco. For EC keys, release 4.40 or higher is required.
Microsoft Azure CLI	Version 2.87.0 or higher. For EC keys, version 2.19 or higher is required

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 6.5.0 or higher
Utimaco PCI-e HSM	u.trust GP HSM CSe-Series/Se-Series PCI-e with firmware SecurityServer 6.5.0 or higher

Table 5: List of hardware requirements

3.4 Prerequisites

Before you begin, ensure that you have following prerequisites completed:

- Set up and configure the Utimaco HSM. Refer to the u.trust GP HSM documentation to set up the HSM.
- Create and store the MBK onto each HSM. Refer to the u.trust GP HSM documentation to set up the MBK.
- Replace the u.trust GP HSM Default Admin with a new admin user.
- Install and set up the operating system listed in [Tested Versions](#).
- Install and set up the u.trust GP HSM as listed in [Tested Versions](#).

- Set up an admin user, as this is required for installing software.

4 Installing and Configuring Utimaco SecurityServer Software

This section describes the process of configuring the Utimaco PKCS#11 Provider for Azure BYOK.

4.1 On Linux

4.1.1 Download and Install Utimaco Software

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

1. Copy the downloaded software at the appropriate location on the Server.
2. Create an `utimaco` folder under the `/opt` directory and further create 2 directories:
 - a. `/opt/utimaco/bin`
 - b. `/opt/utimaco/lib`

> _ Console

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

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

> _ Console

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

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

›_ Console

```
# cd ~/path_to_application_folder

# cp csadm p11tool2 /opt/utimaco/bin

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

4.1.2 u.trust GP HSM PKCS#11 Configuration

1. Create the directory `/etc/utimaco`. Locate the Utimaco PKCS#11 configuration file in your SecurityServer directory, `Linux/x86-64/Crypto_APIS/PKCS11_R3/sample`. Copy the Utimaco PKCS#11 configuration file `cs_pkcs11_R3.cfg` into the `/etc/utimaco` directory.

›_ Console

```
# mkdir /etc/utimaco

# cd <install directory>/Software/Linux/x86-64/Crypto_APIS/PKCS11_R3/sample
# cp cs_pkcs11_R3.cfg /etc/utimaco

# cd /etc/utimaco
```

2. Edit the `cs_pkcs11_R3.cfg` file and make the appropriate changes to the file.

```
cs_pkcs11_R3.cfg
```

```
[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>
```



This integration is not supported with external keystore.



For more information regarding the commands and command parameters, please check the Utimaco CryptoServer documentation. The device may be a CryptoServer (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, it would be good to 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.

4.1.3 Create SO, User and Initialize a Slot

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

First using `p11tool2` create, the SO or Security Officer and then using the `p11tool2` command initialize the Slot that you want to use, as well as the slot user, as shown below.

```
>_ Console
```

```
# ./p11tool2 slot=<slot_no> Label=<token_label> Login=ADMIN,ADMIN.key  
InitToken=<SO_PIN>
```

```
# ./p11tool2 slot=<slot_no> LoginSO=<SO_PIN> InitPin=<CryptoUser_PIN>
```

4.2 On Windows

On Windows, as part of CryptoServer software installation, `cs_pkcs11_R3.cfg` will be automatically created and will be available under the `C:\ProgramData\Utimaco\PKCS11_R3` folder.

4.2.1 Update cs_pkcs11_R3.cfg

Example Values

cs_pkcs11_R3.cfg

```
[Global]

# 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>
```



This integration is not supported with external keystore.



For more information regarding the commands and command parameters, please check the Utimaco CryptoServer documentation. The device may be a CryptoServer (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, it would be good to 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.

5 Implementing BYOK



Please, make sure that you change the labels in brackets to what suits your use case the best.

Example: Change <resource_group> to AzureKeyVaultGroup

5.1 Azure CLI Installation

The version of the Azure CLI used in this guide is 2.87.0. For more information regarding the most recent release, please see the [Azure CLI Release Notes](#) from Microsoft.

5.1.1 Azure CLI Installation on Linux

1. Import the Microsoft repository key.

```
>_ Console
```

```
> rpm --import https://packages.microsoft.com/keys/microsoft.asc
```

2. For RHEL 9, CentOS Stream 9, Rocky Linux 9, or other RHEL-compatible distributions, add the `packages-microsoft-com-prod` repository.

```
>_ Console
```

```
> dnf install -y https://packages.microsoft.com/config/rhel/9.0/packages-microsoft-prod.rpm
```

```
[root@localhost ~]# sudo dnf install -y https://packages.microsoft.com/config/rhel/9.0/packages-microsoft-prod.rpm
Last metadata expiration check: 0:45:53 ago on Mon 15 Jun 2026 09:54:27 PM PDT.
packages-microsoft-prod.rpm                               30 kB/s | 9.2 kB      00:00
Dependencies resolved.
=====
Package                        Architecture      Version           Repository         Size
=====
Installing:
packages-microsoft-prod       noarch            1.1-2            @commandline       9.2 k
=====
Transaction Summary
=====
Install 1 Package

Total size: 9.2 k
Installed size: 2.5 k
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing :
Installing : packages-microsoft-prod-1.1-2.noarch 1/1
Verifying : packages-microsoft-prod-1.1-2.noarch 1/1
Installed:
packages-microsoft-prod-1.1-2.noarch
Complete!
```

Figure 1 : Microsoft packages

3. Install `azure-cli` with the `dnf` install command.

> _ Console

> `dnf install azure-cli`

```
[root@localhost ~]# sudo dnf install -y azure-cli
Microsoft Production                                     654 B/s | 481 B      00:00
Microsoft Production                                     960 kB/s | 983 B     00:00
Importing GPG key 0xBE1229CF:
  Userid : "Microsoft (Release signing) <gpgsecurity@microsoft.com>"
  Fingerprint: BC52 8686 B50D 79E3 39D3 721C EB3E 94AD BE12 29CF
  From : /etc/pki/rpm-gpg/RPM-GPG-KEY-Microsoft
Microsoft Production                                     11 MB/s | 23 MB      00:02
Last metadata expiration check: 0:00:13 ago on Mon 15 Jun 2026 10:41:27 PM PDT.
Dependencies resolved.
=====
Package                        Architecture      Version           Repository         Size
=====
Installing:
azure-cli                      x86_64            2.87.0-1.el9      packages-microsoft-com-prod 39 M
Installing dependencies:
mpdecimal                      x86_64            2.5.1-3.el9       Atalla_app         85 k
python3.12                     x86_64            3.12.13-2.el9_8   appstream          26 k
python3.12-libs                 x86_64            3.12.13-2.el9_8   appstream          9.2 M
python3.12-pip-wheel            noarch            23.2.1-5.el9      appstream          1.5 M
=====
Transaction Summary
=====
Install 5 Packages

Total download size: 50 M
Installed size: 508 M
Downloading Packages:
(1/5): mpdecimal-2.5.1-3.el9.x86_64.rpm                  2.5 MB/s | 85 kB      00:00
(2/5): python3.12-3.12.13-2.el9_8.x86_64.rpm            81 kB/s | 26 kB      00:00
(3/5): python3.12-pip-wheel-23.2.1-5.el9.noarch.rpm      5.3 MB/s | 1.5 MB     00:00
(4/5): python3.12-libs-3.12.13-2.el9_8.x86_64.rpm       11 MB/s | 9.2 MB     00:00
(5/5): azure-cli-2.87.0-1.el9.x86_64.rpm                 24 MB/s | 39 MB      00:01
Total                                                     26 MB/s | 50 MB      00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing :
Installing : python3.12-pip-wheel-23.2.1-5.el9.noarch 1/1
Installing : mpdecimal-2.5.1-3.el9.x86_64            2/5
Installing : python3.12-3.12.13-2.el9_8.x86_64       3/5
```

Figure 2 : Installing Azure CLI packages

5.1.2 Azure CLI Installation on Windows

1. Download the Azure CLI latest version from the above link and click **Install**.



Figure 3 : Installing Azure CLI

2. Click on the **Finish** button to complete the setup.

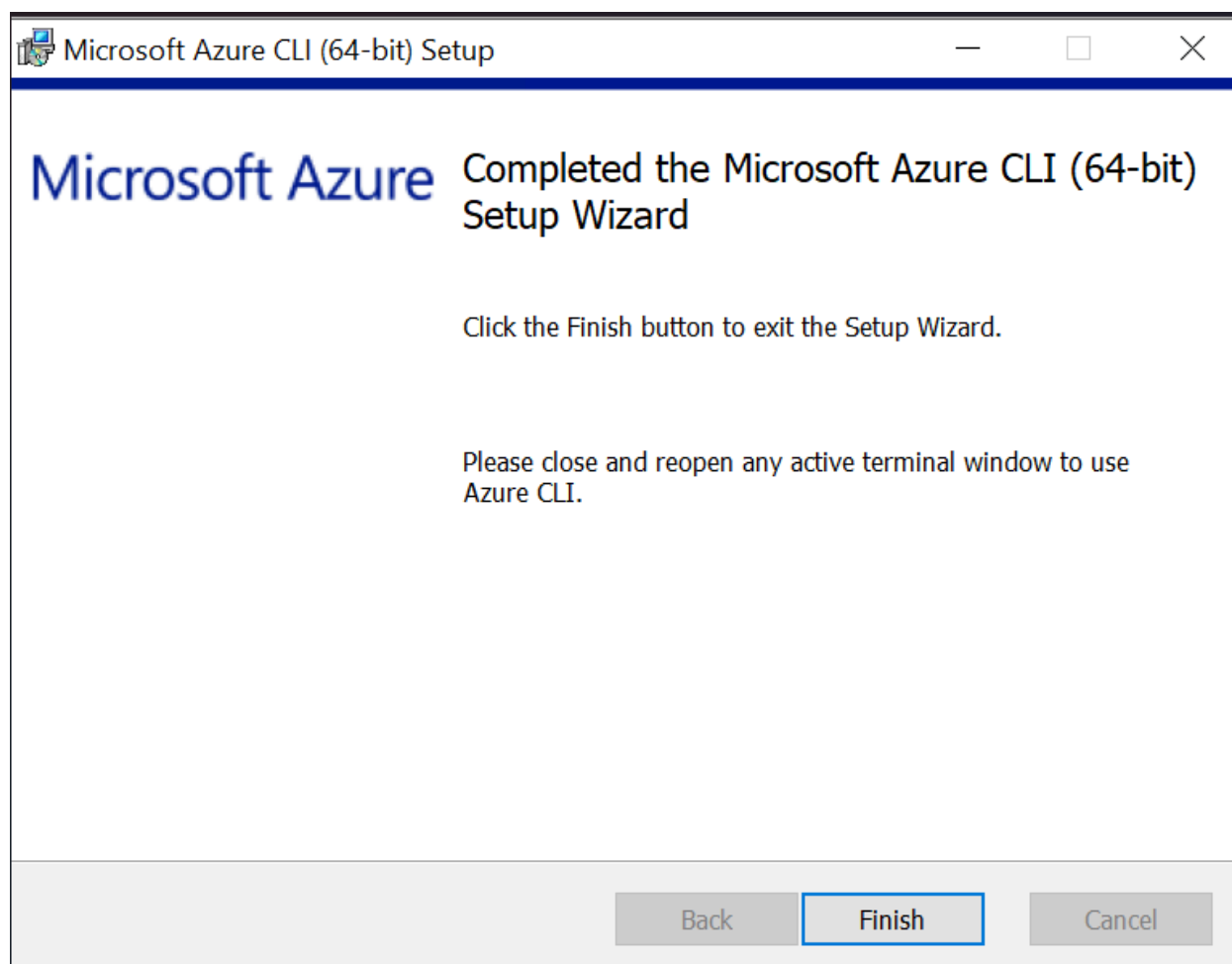


Figure 4 : Azure CLI setup

5.2 Azure Key Vault Premium Subscription

An active Azure subscription with a Premium tier key vault is required to be able to create HSM protected keys.

5.3 Download Utimaco byoktool

To simplify the key export and import process of tenant keys, Utimaco has created a Bring Your Own Key (BYOK) tool. Please reach out to Utimaco so this tool can be provided to you. The byok tool supports all key types (PKCS#11, CNG, JCE, CXI). The storage of keys is still restricted to the internal storage on the Utimaco CryptoServer HSM. The BYOK tool does not support key

creation, only migration. That is why it is important that the settings of the keys' attributes, that you would like to migrate, are set to extractable.

5.4 Creating a Key Vault in Azure

5.4.1 Creating Azure Key Vault with Azure Portal

1. Navigate to portal.azure.com and log in to your Azure Account.
2. Locate the **Key vaults** Azure service, click on it to get redirected to your key vaults.

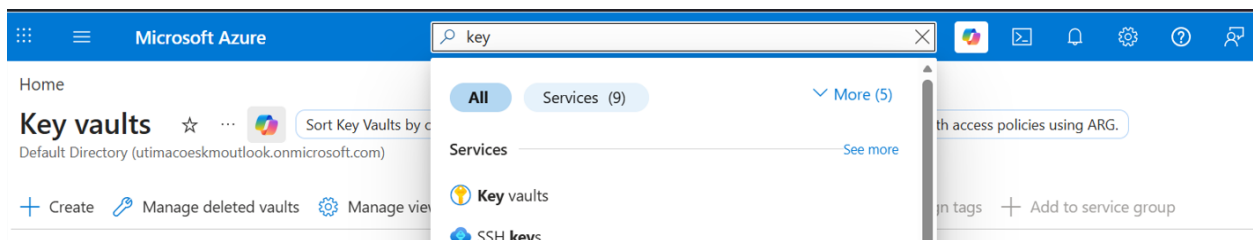


Figure 5 : Azure services

3. Click on **+ Create** to start the process of creating a key vault.

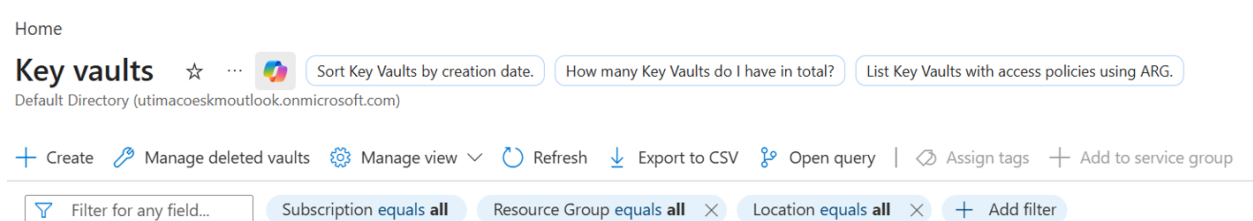


Figure 6 : Home page of Azure Key Vaults

4. Select your subscription and the resource group. Enter the **Key vault name** and **Region** for the key vault you will be using. Make sure to select the **Premium** pricing tier to include support for the HSM backed keys. Set the **Recovery options** according to your company's policies.

Create a key vault ...

Subscription *

Azure subscription 1

Resource group *

igtest

Create new

Instance details

Key vault name * ⓘ

IG-testvault2

Region *

East US

Pricing tier * ⓘ

Premium (includes support for HSM backed keys)

Recovery options

Soft delete protection will automatically be enabled on this key vault. This feature allows you to recover or permanently delete a key vault and secrets for the duration of the retention period. This protection applies to the key vault and the secrets stored within the key vault.

To enforce a mandatory retention period and prevent the permanent deletion of key vaults or secrets prior to the retention period elapsing, you can turn on purge protection. When purge protection is enabled, secrets cannot be purged by users or by Microsoft.

Soft-delete ⓘ

Enabled

Days to retain deleted vaults * ⓘ

90

[Previous](#)[Next](#)[Review + create](#)

Figure 7 : Create key vault page

5. Enable access to your selected users and administer the policy which is the most suitable for your company.

Home > Key vaults

Create a key vault

Basics **Access configuration** Networking Tags Review + create

Configure data plane access for this key vault

To access a key vault in data plane, all callers (users or applications) must have proper authentication and authorization. Authentication establishes the identity of the caller. Authorization determines which operations the caller can execute. [Learn more](#)

Permission model

Grant data plane access by using a [Azure RBAC](#) or [Key Vault access policy](#)

☒ Azure role-based access control (recommended) ⓘ

☐ Vault access policy ⓘ

Resource access

☐ Azure Virtual Machines for deployment ⓘ

☐ Azure Resource Manager for template deployment ⓘ

☐ Azure Disk Encryption for volume encryption ⓘ

Figure 8 : Create key vault access policy page

6. Select your connectivity method.

Home > Key vaults

Create a key vault

Basics Access configuration **Networking** Tags Review + create

You can connect to this key vault either publicly, via public IP addresses or service endpoints, or privately, using a private endpoint.

Enable public access ☒

Public Access

Allow access from:

☒ All networks

☐ Selected networks

ⓘ Traffic from all public networks can access this resource. This is not recommended for private applications or environments. [Learn more about key vault network security configuration](#)

Private endpoint

Create a private endpoint to allow a private connection to this resource. Additional private endpoint connections can be created within the key vault or private link center. [Learn more](#)

Figure 9 : Create key vault networking page

7. If needed, add tags to categorize resources and view consolidated billing.

Home > Key vaults

Create a key vault ...

Basics Access configuration Networking Tags Review + create

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups.

Name ⓘ	Value ⓘ	Resource
	:	Key vault

Figure 10 : Create key vault tags page

- 8. Review the creation of the key vault. After reviewing, click on **Create** to finish the creation of the key vault.

Create a key vault

BasicsAccess configurationNetworkingTagsReview + create

Review + create

Basics

Subscription	Azure subscription 1
Resource group	igtest
Key vault name	IG-testvault2
Region	East US
Pricing tier	Premium
Soft-delete	Enabled
Purge protection during retention period	Disabled
Days to retain deleted vaults	90 days

Access configuration

Azure Virtual Machines for deployment	Disabled
Azure Resource Manager for template deployment	Disabled
Azure Disk Encryption for volume encryption	Disabled

PreviousNextCreate

Figure 11 : Create key vault review and create page

5.4.2 Creating Azure Key Vault on Linux

1. Open the command line interpreter and log in to Azure with the following command:

```
> _ Console
```

```
# az login
```

```
[root@localhost ~]# az login
To sign in, use a web browser to open the page https://login.microsoft.com/device and enter the code FXVDYTM7H to authenticate.
Retrieving tenants and subscriptions for the selection...
[Tenant and subscription selection]
No      Subscription name      Subscription ID      Tenant
-----
[1] *   Azure subscription 1  [REDACTED]          Default Directory

The default is marked with an '*'; the default tenant is 'Default Directory' and subscription is 'Azure subscription 1' ([REDACTED]).
Select a subscription and tenant (Type a number or Enter for no changes):
Tenant: Default Directory
Subscription: Azure subscription 1 [REDACTED]

[Announcements]
With the new Azure CLI login experience, you can select the subscription you want to use more easily. Learn more about it and its configuration at https://go.microsoft.com/fwlink/?linkid=2271236
If you encounter any problem, please open an issue at https://aka.ms/azclibug
[Warning] The login output has been updated. Please be aware that it no longer displays the full list of available subscriptions by default.
[root@localhost ~]#
```

Figure 12 : Azure login

2. To create the Azure Key Vault, you will need to create a resource group. Use the following command to create your own resource group.

> _ Console

```
# az group create --location "<location>" --name "<resource_group>"
```

```
[root@localhost ~]# az group create --location eastus --name igtest
{
  "id": "/subscriptions/[REDACTED]/resourceGroups/igtest",
  "location": "eastus",
  "managedBy": null,
  "name": "igtest",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
[root@localhost ~]#
```

Figure 13 : Creating resource group



For more information regarding the commands and command parameters, please check the Microsoft Azure CLI documentation.

3. Create an Azure Key Vault with premium SKU.

› Console

```
# az keyvault create --location "<location>" --name "<keyvault>"  
--resource-group "<resource_group>" --sku premium
```

```
[root@localhost ~]# az keyvault create --location "eastus" --name "IG-testvault1" --resource-group "igtest" --sku premium
{
  "id": "[redacted]",
  "location": "eastus",
  "name": "IG-testvault1",
  "properties": {
    "accessPolicies": [],
    "createMode": null,
    "enablePurgeProtection": null,
    "enableRbacAuthorization": true,
    "enableSoftDelete": true,
    "enabledForDeployment": false,
    "enabledForDiskEncryption": false,
    "enabledForTemplateDeployment": false,
    "hsmPoolResourceId": null,
    "networkAcls": null,
    "privateEndpointConnections": null,
    "provisioningState": "Succeeded",
    "publicNetworkAccess": "Enabled",
    "sku": {
      "family": "A",
      "name": "premium"
    },
    "softDeleteRetentionInDays": 90,
    "tenantId": "[redacted]",
    "vaultUri": "https://ig-testvault1.vault.azure.net/"
  },
  "resourceGroup": "igtest",
  "systemData": {
    "createdAt": "2026-06-17T11:03:01.461000+00:00",
    "createdBy": "[redacted]",
    "createdByType": "User",
    "lastModifiedAt": "2026-06-17T11:03:01.461000+00:00",
    "lastModifiedBy": "[redacted]",
    "lastModifiedByType": "User"
  },
  "tags": {},
  "type": "Microsoft.KeyVault/vaults"
}
```

Figure 14 : Creating Azure Key Vault

5.4.3 Creating Azure Key Vault on Windows

1. Open the command line interpreter and log in with the following command:

- › Console

```
> az login
```

```
C:\Users\Utimaco>az login
Select the account you want to log in with. For more information on login with Azure CLI, see https://go.microsoft.com/fwlink/?linkid=2271136

Retrieving tenets and subscriptions for the selection...

[Tenant and subscription selection]

No.    Subscription name    Subscription ID    Tenant
-----
[1] *  Azure subscription 1    [redacted]    Default Directory

The default is marked with an *; the default tenant is 'Default Directory' and subscription is 'Azure subscription 1' ([redacted]).

Select a subscription and tenant (Type a number or Enter for no changes):

Tenant: Default Directory
Subscription: Azure subscription 1 [redacted]

[Announcements]
With the new Azure CLI login experience, you can select the subscription you want to use more easily. Learn more about it and its configuration at https://go.microsoft.com/fwlink/?linkid=2271236

If you encounter any problem, please open an issue at https://aka.ms/azclibug

[Warning] The login output has been updated. Please be aware that it no longer displays the full list of available subscriptions by default.
```

Figure 15 : Azure login

2. To create an Azure Key Vault, you will need to create a resource group. Use the following command to create your own resource group.

> _ Console

```
> az group create --location "<location>" --name "<resource_group>"
```

```
C:\Users\Utimaco>az group create --location "eastus" --name "igtest2"
{
  "id": "/subscriptions/[redacted]/resourceGroups/igtest2",
  "location": "eastus",
  "managedBy": null,
  "name": "igtest2",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

Figure 16 : Creating resource group



For more information regarding the commands and command parameters, please check the Microsoft Azure CLI documentation.

3. Create an Azure Key Vault with premium SKU.

> _ Console

```
> az keyvault create --location "<location>" --name "<keyvault>"  
  
--resource-group "<resource_group>" --sku premium
```

```
C:\Users\Utimaco>az keyvault create --location "eastus" --name "IG-testvault3" --resource-group "igtest2" --sku premium  
{  
  "id": "/subscriptions/[redacted]resourceGroups/igtest2/providers/Microsoft.KeyVault/vaults/IG-testvault3",  
  "location": "eastus",  
  "name": "IG-testvault3",  
  "properties": {  
    "accessPolicies": [],  
    "createMode": null,  
    "enablePurgeProtection": null,  
    "enableRbacAuthorization": true,  
    "enableSoftDelete": true,  
    "enabledForDeployment": false,  
    "enabledForDiskEncryption": false,  
    "enabledForTemplateDeployment": false,  
    "hsmPoolResourceId": null,  
    "networkAcls": null,  
    "privateEndpointConnections": null,  
    "provisioningState": "Succeeded",  
    "publicNetworkAccess": "Enabled",  
    "sku": {  
      "family": "A",  
      "name": "premium"  
    },  
    "softDeleteRetentionInDays": 90,  
    "tenantId": "[redacted]",  
    "vaultUri": "https://ig-testvault3.vault.azure.net/"  
  },  
  "resourceGroup": "igtest2",  
  "systemData": {  
    "createdAt": "2026-06-29T11:29:01.267000+00:00",  
    "createdBy": "[redacted]",  
    "createdByType": "User",  
    "lastModifiedAt": "2026-06-29T11:29:01.267000+00:00",  
    "lastModifiedBy": "[redacted]",  
    "lastModifiedByType": "User"  
  },  
}
```

Figure 17 : Creating Azure Key Vault

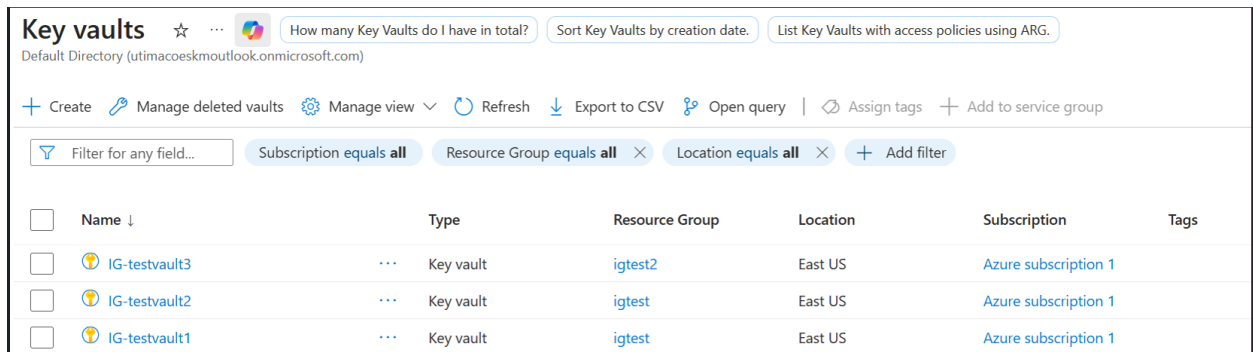
5.5 Generating Key Exchange Key (KEK)

The KEK (Key Exchange Key) is an RSA key, generated in the Key Vault. The KEK must be:

1. An RSA-HSM key (2048-bit or 3072-bit or 4096-bit).
2. Generated in the same key vault where you intend to import the tenant key to.
3. Created with the allowed key operations set to import.

5.5.1 Creating a KEK with Azure Portal

To create a key within your recently created key vault, click on the name of your Azure **Key vault** and follow the next steps.



Key vaults						
Default Directory (utimacoeskmountlook.onmicrosoft.com)						
+ Create Manage deleted vaults Manage view Refresh Export to CSV Open query Assign tags Add to service group						
Filter for any field... Subscription equals all Resource Group equals all Location equals all Add filter						
☐	Name ↓	Type	Resource Group	Location	Subscription	Tags
☐	IG-testvault3	Key vault	igtest2	East US	Azure subscription 1	
☐	IG-testvault2	Key vault	igtest	East US	Azure subscription 1	
☐	IG-testvault1	Key vault	igtest	East US	Azure subscription 1	

Figure 18 : Key vaults home page

1. Under the **Settings** menu select the **Keys** setting.

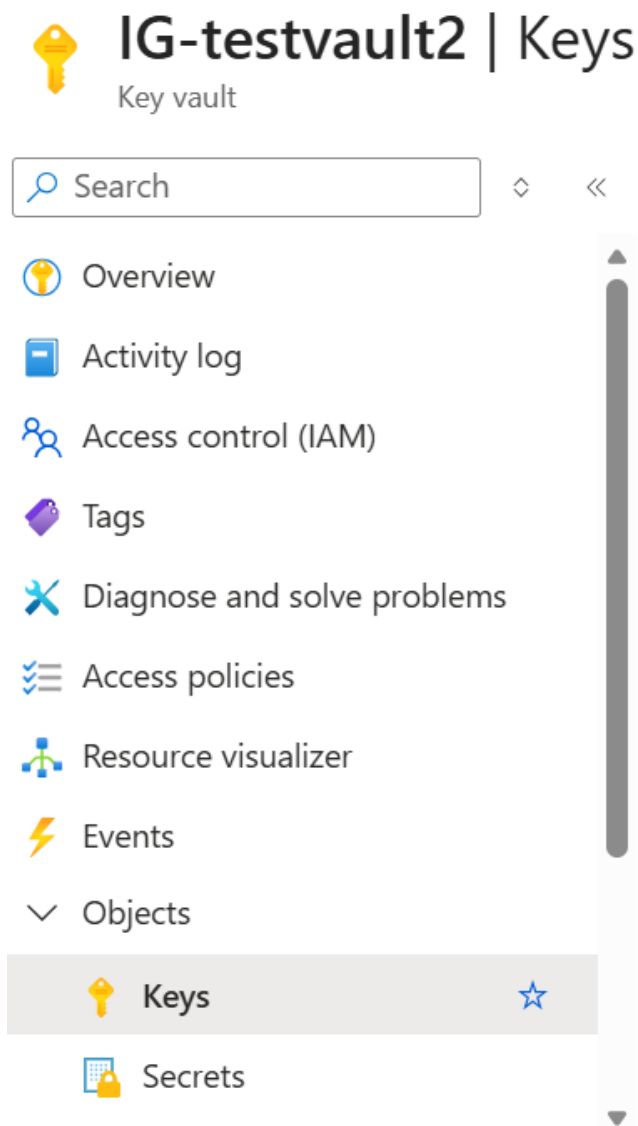


Figure 19 : Key vault menu

2. Click on **Generate/Import**.

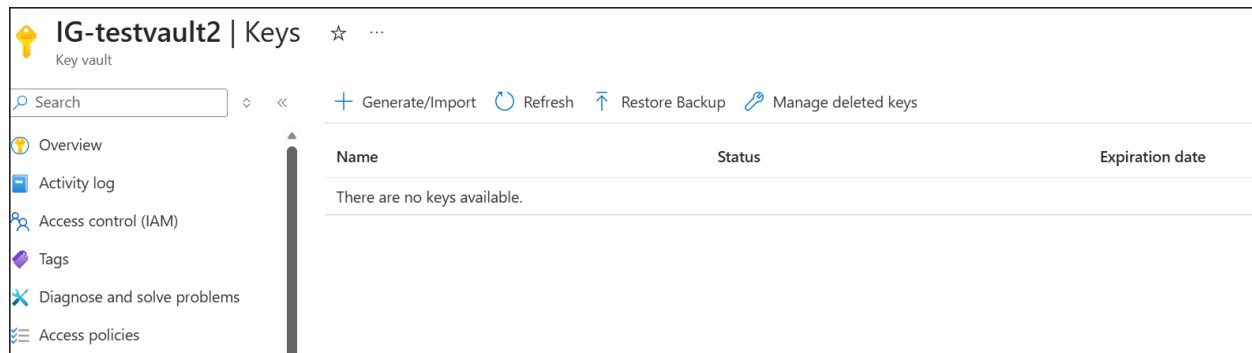


Figure 20 : Default key vault keys setting page

3. In the **Options** drop-down menu select **Generate Key Encryption Key for importing HSM-protected Keys**, add a name to the key and select the RSA key size. If needed, set the activation and expiration date for the key.

Home > Key vaults > IG-testvault2 | Keys

Create a key

Options: Generate Key Encryption Key for Importing HSM-protected Keys

Name: utimacotestKeyRSA2

Key type: RSA-HSM

RSA key size: 2048

Set activation date: ☒

Activation date: 06/29/2026 7:55:28 PM (UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi

Set expiration date: ☒

Expiration date: 07/01/2026 7:55:28 PM (UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi

Enabled: ☒ Yes ☐ No

Figure 21 : Example of create a key page

4. After the key is created it will be display under the key vault created and being used. Navigate to the following path to see the newly generated key.

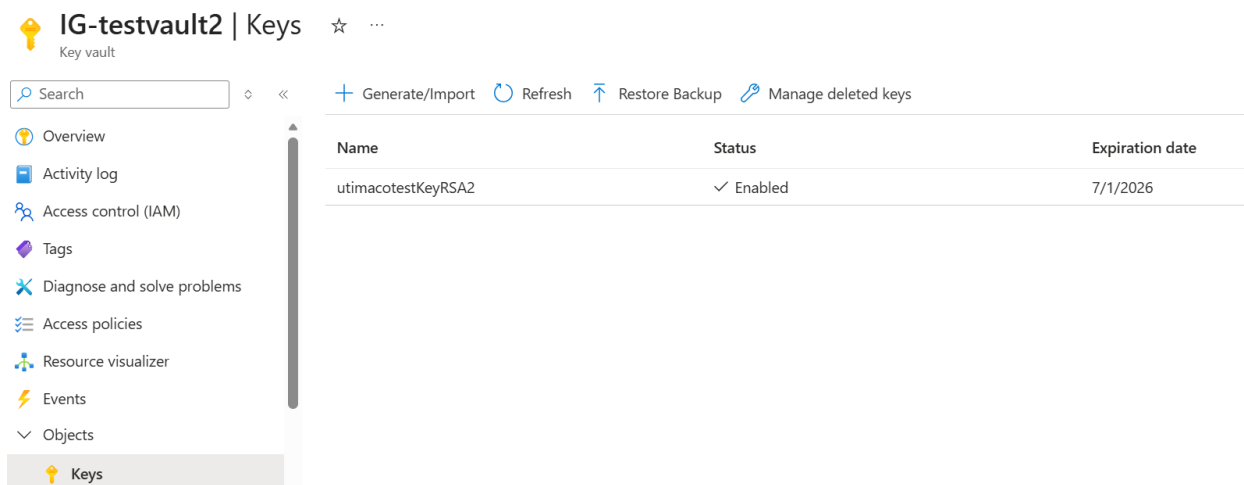


Figure 22 : Generated keys

5. The key we just created has an identifier which will be needed in the next steps. To find this **Key Identifier**, click on the newly created key in your **Key Vault** to display its properties. The **Key Identifier** will be needed in the steps for *Generating and Preparing your Tenant Key*.

Home > Key vaults > IG-testvault2 | Keys > utimacotestKeyRSA2

 **0327e357763e4bde96093f60139b5c80**  ...

Key Version

 Save  Discard changes  Download public key

Properties

Key type	RSA-HSM
RSA key size	2048
Created	6/29/2026, 7:57:12 PM
Updated	6/29/2026, 7:57:12 PM
Key Identifier	https://ig-testvault2.vault.azure.net/keys/utimac... 

Settings

Set activation date ⓘ ☒

Activation date 
 (UTC+05:30) Chennai, Kolkata, Mumbai, New ... 

Set expiration date ⓘ ☒

Figure 23 : Key properties

5.5.2 Creating a KEK on Linux

1. Create a KEK with the key operations set to import. The KEK can be an RSA key of different sizes such as: 2048-bit, 3072-bit or 4096-bit. It is advisable to create a key with the length suitable for your use case.

> _ Console

```
# az keyvault key create --name "<keyvault_key>" --vault-name "<keyvault>"
-- kty RSA-HSM --size 2048 --ops import
```

```
[root@localhost ~]# az keyvault key create --name "utimacotestKeyRSA1" --vault-name "IG-testvault1" --key RSA-HSM --size 2048 --ops import
{
  "attributes": {
    "created": "2026-06-29T10:37:13+00:00",
    "enabled": true,
    "expires": "2026-07-01T10:37:13+00:00",
    "exportable": false,
    "hsmPlatform": "2",
    "notBefore": null,
    "recoverableDays": 90,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2026-06-29T10:37:13+00:00"
  },
  "key": {
    "crv": null,
    "d": null,
    "dp": null,
    "dq": null,
    "e": "AQAB",
    "k": null,
    "keyOps": [
      "import"
    ],
    "kid": "https://ig-testvault1.vault.azure.net/keys/utimacotestKeyRSA1/dd708eacba01485cb9f86c829f99457b",
    "kty": "RSA-HSM",
    "n": "yIQ2xUjV371bP0NNft1SS63Ke0T/pYNaSJS9VQszHvnoT6lPKgJe1A7eB3pLhmMNLNp6lwaOkNoiVZe8QnfdfiCWxcuKvnLsmiO43wH4lLmjo+kBalAezCPAWNyHHkntWeBTZYfg3CogSmO30ga01br4FKCT3T7udxT2RFTSarkqJ7A2/PSVnW29Zum4JohY3/ttAeLJym0yELWn6/tIL2KkYMiJGvTUGwYooIST/TitvH8B1AMwxAuwlmb9LzuBtqdiKunb0jaNcfzwxlrmSo22mi+eUb16QYkWOYqJhWxz7PsUgJzYYUI7LEKyE8fia5tjd8jmYs2tcvo3Hw==",
    "p": null,
    "q": null,
    "qi": null,
    "t": null,
    "x": null,
    "y": null
  },
  "managed": null,
  "releasePolicy": null,
  "tags": null
}
```

Figure 24 : Creating KEK



After the command has been successfully executed, please make sure to note down the key identifier in the command printout as it will be used when wrapping your tenant key.

5.5.3 Creating a KEK on Windows

1. Create a KEK with the key operations set to import. The KEK can be an RSA key of different sizes such as: 2048-bit, 3072-bit or 4096-bit. It is advisable to create a key with the length suitable for your use case.

> _ Console

```
> az keyvault key create --name "<keyvault_key>" --vault-name "<keyvault>"
-- kty RSA-HSM --size 2048 --ops import
```

```
C:\Users\Utimaco>az keyvault key create --name "utimacotestKeyRSA3" --vault-name "IG-testvault3" --key-type RSA-HSM --size 2048 --ops import
{
  "attributes": {
    "created": "2026-06-29T11:47:06+00:00",
    "enabled": true,
    "expires": "2026-07-01T11:47:06+00:00",
    "exportable": false,
    "hsmPlatform": "2",
    "notBefore": null,
    "recoverableDays": 90,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2026-06-29T11:47:06+00:00"
  },
  "key": {
    "crv": null,
    "d": null,
    "dp": null,
    "dq": null,
    "e": "AQAB",
    "k": null,
    "keyOps": [
      "import"
    ],
    "kid": "https://ig-testvault3.vault.azure.net/keys/utimacotestKeyRSA3/4cb1d670629444a28807a64bf18195b4",
    "keyType": "RSA-HSM",
    "n": "tdR2uGj+qLNeVF+Yp1suGta0xd7pgnSoiUF0f04aNX4Mi9o6+YVd0czsdK5QIT1waOfxiU7f5+HT7BRdLJMXUH0iCa+9wQf0CkwwJ/aknecah6vj+3kcyNe4ZqC2PojmYPSsr0Yi5YTQnbWxOAKR1cbW30nmY8b
bC5b3jSp0GvaQ40CoN8FzFiQI4XKLux3yAwVcko8Y5Qotr27GPQ0D68dNFK8DHW+wk/HQYxcv1I84757t36W/1rUkUPTTF/HxzmCa72rh05kNB+6HfPc7x0ZfhP3STt/1Xuo+3VbDAALZusfw31p0q+9+TYxmsbi06DNQdh
hyNhhUTYv==",
    "p": null,
    "q": null,
    "qi": null,
    "t": null,
    "x": null,
    "y": null
  },
  "managed": null,
}
```

Figure 25 : Creating KEK on Windows



After the command has been successfully executed, please make sure to note down the key identifier in the command printout as it will be used when wrapping your tenant key.

5.6 Downloading KEK Public Key

You will need to download the Public Key of the Key Exchange Key generated on the Azure Key Vault. The Public Key of the KEK will be used to encrypt your tenant key.

5.6.1 Downloading KEK Public Key with Azure Portal

To download the KEK public key, navigate to your Azure Portal. Go to your **Key vaults** and select the key you would like to download. Click on **Download public key**.

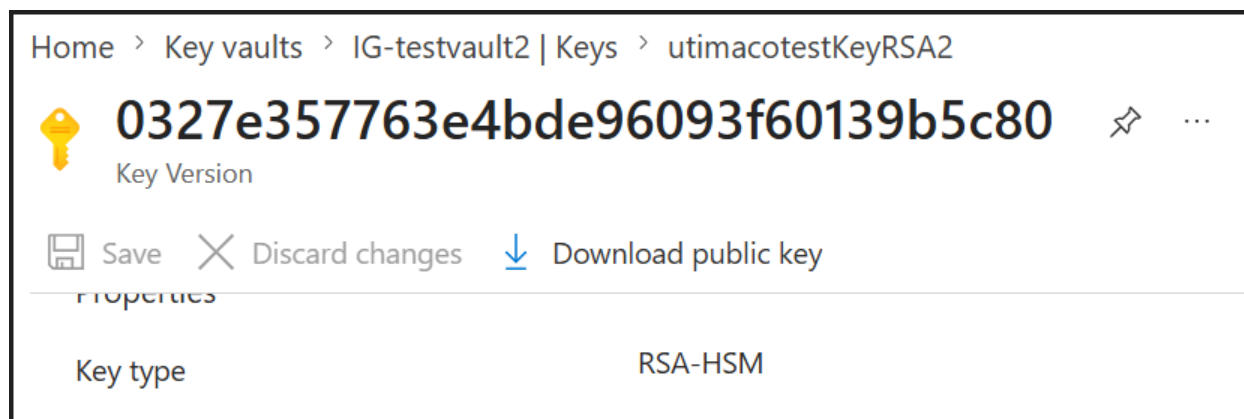


Figure 26 : Key properties

5.6.2 Downloading KEK Public Key on Linux

Execute the following command to download the KEK public key in the `*.pem` format.

```
> _ Console

# az keyvault key download --name "<keyvault_key>" --vault-name
"<keyvault>" --file <keyvault_key>.publickey.pem
```

```
[root@localhost ~]# az keyvault key download --name "utimacotestKeyRSA1" --vault-name "IG-testvault1" --file utimacotestKeyRSA1.publickey.pem
[root@localhost ~]#
[root@localhost ~]# ls
anaconda-ks.cfg  Azure-BYOKECtest  byoktool  cxi.log  etcd-backup.db  etcd-tar.gz  etcd-v3.5.0-linux-amd64  index.html  index.html.1  k8s-rpms  pyclient.py  rpmbuild  sybinit.err  utimacotestKeyRSA1.publickey.pem
[root@localhost ~]#
```

Figure 27 : Download key

5.6.3 Downloading KEK Public Key on Windows

Execute the following command to download the KEK public key in the `*.pem` format.

```
>_ Console
```

```
# az keyvault key download --name "<keyvault_key>" --vault-name  
"<keyvault>" --file <keyvault_key>.publickey.pem
```

```
C:\Users\Utimaco>az keyvault key download --name "utimacotestKeyRSA3" --vault-name "IG-testvault3" --file utimacotestKeyRSA3.publickey.pem
```

Figure 28 : Download key

5.7 Generating and Preparing your Tenant Key

Ensure that you have created an HSM user that can manage crypto operations (CryptoUser). The byoktool supports all key types (PKCS#11, CNG, JCE, CXI). In this guide we will create a PKCS#11 key, see [CSPKCS11RM] for more information. For other types, refer to the documentation provided to you on the Utimaco Product CD.

5.7.1 Generating Tenant Key on Linux

1. Use the following command to generate an RSA tenant key:

```
>_ Console
```

```
# p11tool2 slot=<slot_no.> loginuser=<user_password>  
PubKeyAttr=CKA_LABEL="<tenant_public_key>",CKA_MODULUS_BITS=<keysize>  
PrvKeyAttr=CKA_LABEL="<tenant_private_key>",CKA_EXTRACTABLE=CK_TRUE  
GenerateKeyPair=RSA
```

```
[root@localhost bin]# ./p11tool2 slot=0 loginuser=12345678 PubKeyAttr=CKA_LABEL="tenant_public_RSAkey",CKA_MODULUS_BITS=2048 PrvKeyAttr=CKA_LABEL="tenant_private_RSAkey",CKA_EXTRACTABLE=CK_TRUE GenerateKeyPair=RSA
[root@localhost bin]#
[root@localhost bin]# ./p11tool2 LoginUser=12345678 ListObjects

CKO_PUBLIC_KEY:
+ 1.1
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = FD8A6B38-D90A-4BE7-AE80-46113CB91F6E
  CKA_LABEL              = tenant_public_RSAkey
  CKA_ID                 =

CKO_PRIVATE_KEY:
+ 2.1
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = E510F733-E932-4620-A60F-18F45391D8D6
  CKA_SENSITIVE           = CK_TRUE
  CKA_EXTRACTABLE        = CK_TRUE
  CKA_LABEL              = tenant_private_RSAkey
  CKA_ID                 =
```

Figure 29 : List newly generated RSA keys

<keysize> is the RSA key size in bits. Azure supports 2048, 3072 and 4096. Use the following command to generate an EC tenant key:

>_ Console

```
# p11tool2 slot=<slot_no.> loginuser=<user_password>
PubKeyAttr=CKA_LABEL="<tenant_public_key>",CKA_EC_PARAMS=oid:<curvename>
PrvKeyAttr=CKA_LABEL="<tenant_private_key>",CKA_EXTRACTABLE=CK_TRUE
GenerateKeyPair=ECC
```

<curvename> is the name of the EC curve. For Azure, this needs to be NIST-P256, NIST- P384 or NIST-P521.


```
[root@localhost bin]# ./p11tool2 slot=0 loginuser=12345678 PubKeyAttr=CKA_LABEL="tenant_public_ECCkey",CKA_EC_PARAMS=oid:NIST-P256 PrvKeyAttr=CKA_LABEL="tenant_private_ECCkey",CKA_EXTRACTABLE=CK_TRUE GenerateKeyPair=ECC
[root@localhost bin]#
[root@localhost bin]# ./p11tool2 LoginUser=12345678 ListObjects

CKO_PUBLIC_KEY:
+ 1.1
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = FD8A6B38-D90A-4BE7-AE80-46113CB91F6E
  CKA_LABEL              = tenant_public_RSAkey
  CKA_ID                 =
+ 1.2
  CKA_KEY_TYPE           = CKK_ECDSA
  CKA_UNIQUE_ID          = DED21c7D-0E29-48F0-9B8E-3A15143893AF
  CKA_LABEL              = tenant_public_ECCkey
  CKA_ID                 =

CKO_PRIVATE_KEY:
+ 2.1
  CKA_KEY_TYPE           = CKK_ECDSA
  CKA_UNIQUE_ID          = 98EFD0A0-322E-42EB-9D65-1F20E7E1363C
  CKA_SENSITIVE          = CK_TRUE
  CKA_EXTRACTABLE        = CK_TRUE
  CKA_LABEL              = tenant_private_ECCkey
  CKA_ID                 =
+ 2.2
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = E510F133-E932-4620-A60F-18F45391D8D6
  CKA_SENSITIVE          = CK_TRUE
  CKA_EXTRACTABLE        = CK_TRUE
  CKA_LABEL              = tenant_private_RSAkey
  CKA_ID                 =
```

Figure 30 : List newly generated ECC keys



The "oid:<curvename>" syntax is supported starting with SecurityServer 4.40.



The attribute CKA_EXTRACTABLE must be set to CK_TRUE.

- Navigate to the folder where you have the `byoktool` saved. Execute the following command to wrap the tenant key by using the `KeyVaultKey`, downloaded from the Azure Key Vault.

> _ Console

```
# byoktool Dev=<IP_of_UTIMACO_HSM> LogonPass=<user>,<user_password>
Label="<tenant_private_key>" CSP=azure
PublicKey="<keyvaultkey>.publickey.pem" KID="<kid>"

WrappedKey="<wrappedkey>"
```

For the RSA Key

```
[root@localhost ~]# ./byoktool Dev=3001@172.31.1.37 LogonPass=USR_0000,12345678 Label="tenant_private_RSAkey" CSP=azure PublicKey="utimacotestKeyRSA1.publickey.pem" KID="https://ig-testvault1.vault.azure.net/keys/utimacotestKeyRSA1/dd708eacba01485cb9f86c829f99457b" WrappedKey="Azure-BYOKRSAtest"
byoktool 1.0.0 - (c) Utimaco
Logging in USR_0000
Finding key to be wrapped
Reading public key
Performing key wrap
Writing output
```

Figure 31 : Wrap the tenant key for RSA

For the ECC Key

```
[root@localhost ~]# ./byoktool Dev=3001@172.31.1.37 LogonPass=USR_0000,12345678 Label="tenant_private_ECCkey" CSP=azure PublicKey="utimacotestKeyECC1.publickey.pem" KID="https://ig-testvault1.vault.azure.net/keys/utimacotestKeyECC1/f5297281e50e4709b7610e234ff088e6" WrappedKey="Azure-BYOECCtest"
byoktool 1.0.0 - (c) Utimaco
Logging in USR_0000
Finding key to be wrapped
Reading public key
Performing key wrap
Writing output
```

Figure 32 : Wrap the tenant key for ECC

Command Parameters:

- **<user>** is "USR_0013" for PKCS#11 slot 13. Any other user/password combination with access to the tenant_private_key is possible as well.
- **<keyvaultkey>** is the filename of the public key downloaded from Azure key vault.
- **<kid>** is the key identifier of the KEK in Key Vault (e.g., https://KeyVaultUtimacoHSM.vault.azure.net/keys/mykek/eba63d27es214e028839s77fc905621).
- **<wrappedkey>** is the filename of the wrapped tenant key.

5.7.2 Generating Tenant Key on Windows

1. Use the following command to generate an RSA tenant key:

> _ Console

```
> p11tool2 slot=<slot_no.> loginuser=<user_password>
PubKeyAttr=CKA_LABEL="<tenant_public_key>",CKA_MODULUS_BITS=<keysize>
PrvKeyAttr=CKA_LABEL="<tenant_private_key>",CKA_EXTRACTABLE=CK_TRUE
GenerateKeyPair=RSA
```

```

C:\Users\Utimaco>p11tool2 slot=0 loginuser=12345678 PubKeyAttr=CKA_LABEL="tenant_public_RSAkey",CKA_MODULUS_BITS=2048 PrvKeyAttr=CKA_LABEL="tenant_private_RSAkey",CKA_EXTRACTABLE=CK_TRUE GenerateKeyPair=RSA

C:\Users\Utimaco>p11tool2 LoginUser=12345678 ListObjects

CKO_PUBLIC_KEY:

+ 1.1
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = B234408BC-D4B6-499E-90CB-FFB958F3DD8F
  CKA_LABEL              = tenant_public_RSAkey
  CKA_ID                 =

CKO_PRIVATE_KEY:

+ 2.1
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = 50C447A5-FBA5-4606-819D-65F43321204D
  CKA_SENSITIVE          = CK_TRUE
  CKA_EXTRACTABLE        = CK_TRUE
  CKA_LABEL              = tenant_private_RSAkey
  CKA_ID                 =

```

Figure 33 : List newly generated RSA keys

`<keysize>` is the RSA key size in bits. Azure supports 2048, 3072 and 4096. Use the following command to generate an EC tenant key:

> _ Console

```

> p11tool2 slot=<slot_no.> loginuser=<user_password>
PubKeyAttr=CKA_LABEL="<tenant_public_key>",CKA_EC_PARAMS=oid:<curvename>
PrvKeyAttr=CKA_LABEL="<tenant_private_key>",CKA_EXTRACTABLE=CK_TRUE
GenerateKeyPair=ECC

```

`<curvename>` is the name of the EC curve. For Azure, this needs to be NIST-P256, NIST- P384 or NIST-P521.

```

C:\Users\Utimaco>pl1tool2 slot=0 loginuser=12345678 PubKeyAttr=CKA_LABEL="tenant_public_ECCKey",CKA_EC_PARAMS=oid:NIST-P256 PrvKeyAttr=CKA_LABEL="tenant_private_ECCKey",CKA_EXTRACTABLE=CK_TRUE GenerateKeyPair=ECC

C:\Users\Utimaco>pl1tool2 LoginUser=12345678 ListObjects

CKO_PUBLIC_KEY:

+ 1.1
  CKA_KEY_TYPE           = CKK_RSA
  CKA_UNIQUE_ID          = B234408C-D486-499E-90CB-FFB958F3DD8F
  CKA_LABEL              = tenant_public_RSAkey
  CKA_ID                 =

1.2
  CKA_KEY_TYPE           = CKK_ECDSA
  CKA_UNIQUE_ID          = DABE700F-3085-4EE2-8E58-136576975DCD
  CKA_LABEL              = tenant_public_ECCKey
  CKA_ID                 =

CKO_PRIVATE_KEY:

+ 2.1
  CKA_KEY_TYPE           = CKK_ECDSA
  CKA_UNIQUE_ID          = DF18F0AE-34D3-4A5A-8BF5-B195F933D5F0
  CKA_SENSITIVE          = CK_TRUE
  CKA_EXTRACTABLE        = CK_TRUE
  CKA_LABEL              = tenant_private_ECCKey
  CKA_ID                 =

```

Figure 34 : List newly generated ECC keys



The "oid:<curvename>" syntax is supported starting with SecurityServer 6.5.0.



The attribute CKA_EXTRACTABLE must be set to CK_TRUE

2. Navigate to the folder where you have the `byoktool` saved. Execute the following command to wrap the tenant key by using the `KeyVaultKey`, downloaded from the Azure Key Vault.

> _ Console

```

> byoktool Dev=<IP_of_UTIMACO_HSM> LogonPass=<user>,<user_password>
Label="<tenant_private_key>" CSP=azure
PublicKey="<keyvaultkey>.publickey.pem" KID="<kid>"

WrappedKey="<wrappedkey>"

```

For the RSA Key

```
C:\Users\Utimaco>byoktool Dev=3001@127.0.0.1 LogonPass=USR_0000,12345678 Label="tenant_private_RSAkey" CSP=azure PublicKey="utimacotestKeyRSA3.publickey.pem" KID="https://g-testvault3.vault.azure.net/keys/utimacotestKeyRSA3/4cb1d670629444a28807a64bf18195b4" WrappedKey="Azure-BYOKRSAtest"
byoktool 1.0.0 - (c) Utimaco
Logging in USR_0000
Finding key to be wrapped
Reading public key
Performing key wrap
Writing output
```

Figure 35 : Wrap the tenant key for RSA

For the ECC Key

```
C:\Users\Utimaco>byoktool Dev=3001@127.0.0.1 LogonPass=USR_0000,12345678 Label="tenant_private_ECCkey" CSP=azure PublicKey="utimacotestKeyECC3.publickey.pem" KID="https://g-testvault3.vault.azure.net/keys/utimacotestKeyECC3/ac8953e585c24cc9a0ee291cefb5c4b3" WrappedKey="Azure-BYOKECCTest"
byoktool 1.0.0 - (c) Utimaco
Logging in USR_0000
Finding key to be wrapped
Reading public key
Performing key wrap
Writing output
```

Figure 36 : Wrap the tenant key for ECC

Command Parameters:

- **<user>** is "USR_0020" for PKCS#11 slot 20. Any other user/password combination with access to the tenant_private_key is possible as well.
- **<keyvaultkey>** is the filename of the public key downloaded from Azure key vault.
- **<kid>** is the key identifier of the KEK in Key Vault (e.g., https://KeyVaultUtimacoHSM.vault.azure.net/keys/mykek/eba63d27es214e028839s77fc905621).
- **<wrappedkey>** is the filename of the wrapped tenant key.

5.8 Importing Tenant Key to Azure Key Vault

5.8.1 Importing on Linux

Use the Azure CLI to import the wrapped key to your Azure key vault, generated in the [previous steps](#).

1. Execute the following command to import an RSA tenant key:

>_ Console

```
# az keyvault key import --vault-name <keyvault> --name <WrappedKeyName> --
byok-file "wrappedkey>"
```

```
[root@localhost ~]# az keyvault key import --vault-name IG-testvault1 --name utimacotestKeyRSA1 --byok-file "Azure-BYOKRSatest"
{
  "attributes": {
    "created": "2026-06-29T10:48:33+00:00",
    "enabled": true,
    "expires": null,
    "exportable": false,
    "hsmPlatform": "2",
    "notBefore": null,
    "recoverableDays": 90,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2026-06-29T10:48:33+00:00"
  },
  "key": {
    "crv": null,
    "d": null,
    "dp": null,
    "dq": null,
    "e": "AQAB",
    "k": null,
    "keyOps": [
      "encrypt",
      "decrypt",
      "sign",
      "verify",
      "wrapKey",
      "unwrapKey"
    ],
    "kid": "https://ig-testvault1.vault.azure.net/keys/utimacotestKeyRSA1/dbe9da648fc14ef2a3eeb6d71ca68959",
    "kty": "RSA-HSM",
    "n": "guCL37yfOfC2tIxcPjn4Ga9JlDxWgUaWH9913e/XGwuWUTwOSMZe5ec8PAecvflGCL5u8hnoJkqfyOGDpnV5glgeIK27ITbommjfcghI4+UsOkQszmZfbXDy8xi68yTEnK0TRyyeWTU+l3e4D0Y9PHRIm1zh9OakHDwMmJPb9JQWC5sTT0mcKjSFmni/Io0VzPaojhuNiSVYvanu+tmW188Sggazv89pDgnaXV/Y2L3OKXJbk3SpJNwc0OpsWfdkiOyVja4+HdcHjhr/QyDMMNf+e14+D3r60ntx5xE6wL1tQ9t",
    "p": null,
    "q": null,
    "qi": null,
    "t": null,
    "x": null,
    "y": null
  },
  "managed": null,
  "releasePolicy": null,
  "tags": null
}
```

Figure 37 : List importing tenant key

Execute the following command to import an EC tenant key:

> _ Console

```
# az keyvault key import --vault-name <keyvault> --name <WrappedKeyName> --byok-file "wrappedkey" --kty EC --curve <curvename>
```

For **<curvename>** Azure supports P-256, P-384 and P-521. The curve name must match the actual key.

```
[root@localhost ~]# az keyvault key import --vault-name IG-testvault1 --name utimacotestKeyECC1 --byok-file "Azure-BYOKECCTest" --kty EC --curve P-256
{
  "attributes": {
    "created": "2026-06-29T11:02:34+00:00",
    "enabled": true,
    "expires": null,
    "exportable": false,
    "hsmPlatform": "2",
    "notBefore": null,
    "recoverableDays": 90,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2026-06-29T11:02:34+00:00"
  },
  "key": {
    "crv": "P-256",
    "d": null,
    "dp": null,
    "dq": null,
    "e": null,
    "k": null,
    "keyOps": [
      "sign",
      "verify"
    ],
    "kid": "https://ig-testvault1.vault.azure.net/keys/utimacotestKeyECC1/c3fe020d35c14fb49fdeafdbfd6c1bac",
    "kty": "EC-HSM",
    "n": null,
    "p": null,
    "q": null,
    "qi": null,
    "t": null,
    "x": "DFvFUcGgNRfXDDJCcOq5VYjwvCiENAlBDEMIiHxh+34=",
    "y": "gB7BF7vo6rs9pmuZxavJhKtaqJ7x7Ls/ej0wmtzah90="
  },
  "managed": null,
  "releasePolicy": null,
  "tags": null
}
```

Figure 38 : List importing tenant key

+ Generate/Import ↻ Refresh ↑ Restore Backup 🔗 Manage deleted keys		
Name	Status	Expiration date
utimacotestKeyECC1	✓ Enabled	7/1/2026
utimacotestKeyRSA1	✓ Enabled	

Figure 39 : List importing tenant key

2. Use the following command to check if the key has been successfully imported to the Azure Key vault:

>_ Console

```
# az keyvault key show --vault-name <keyvault> --name <WrappedKeyName>
```

You can also check if the key is visible on your Azure portal.

For the RSA Key

```
[root@localhost ~]# az keyvault key show --vault-name IG-testvault1 --name utimacotestKeyRSA1
{
  "attributes": {
    "created": "2026-06-29T10:48:33+00:00",
    "enabled": true,
    "expires": null,
    "exportable": false,
    "hsmPlatform": "2",
    "notBefore": null,
    "recoverableDays": 90,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2026-06-29T10:48:33+00:00"
  },
  "key": {
    "crv": null,
    "d": null,
    "dp": null,
    "dq": null,
    "e": "AQAB",
    "k": null,
    "keyOps": [
      "encrypt",
      "decrypt",
      "sign",
      "verify",
      "wrapKey",
      "unwrapKey"
    ],
    "kid": "https://ig-testvault1.vault.azure.net/keys/utimacotestKeyRSA1/dbe9da648fc14ef2a3eeb6d71ca68959",
    "kty": "RSA-HSM",
    "n": "quCL37yOfC2tIxcPjn4Ga9JLDxWgUaWH9913e/XGwuUUTwOSMZe5ec8FAecvflGCL5u8hnoJkqfyOGDpnV5glgeIK27ITbommjfcghI4+UsOkQszmZfbXDy8xi68yTEnK0TRyyeWTU+l3e4DOY9PHRIm1zh9OakHDwMmJPb9JQWC5sTT0mcKjSFmni/Io0VzPaoJhuNiSVYvanu+tmW188Sggazv89pDgnaXV/Y2L3OKXJbk3SpJNwc0OpsWfdkiOyJja4+HdcHjhr/QyDMMNf+e14+D3r60ntx5xE6wL1tQ9tkqDFJGVSRWOSzstWq3QN12NmGNYNRGwEjltWuQ==",
    "p": null,
    "q": null,
    "qi": null,
    "t": null,
    "x": null,
    "y": null
  },
  "managed": null,
  "releasePolicy": null,
  "tags": null
}
```

Figure 40 : List RSA key

For the ECC Key

```
[root@localhost ~]# az keyvault key show --vault-name IG-testvault1 --name utimacotestKeyECC1
{
  "attributes": {
    "created": "2026-06-29T11:02:34+00:00",
    "enabled": true,
    "expires": null,
    "exportable": false,
    "hsmPlatform": "2",
    "notBefore": null,
    "recoverableDays": 90,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2026-06-29T11:02:34+00:00"
  },
  "key": {
    "crv": "P-256",
    "d": null,
    "dp": null,
    "dq": null,
    "e": null,
    "k": null,
    "keyOps": [
      "sign",
      "verify"
    ],
    "kid": "https://ig-testvault1.vault.azure.net/keys/utimacotestKeyECC1/c3fe020d35c14fb49fdeafdbfd6c1bac",
    "kty": "EC-HSM",
    "n": null,
    "p": null,
    "q": null,
    "qi": null,
    "t": null,
    "x": "DFvFUcGsNRfXDDJC0q5VYjwvCiENAlBDBMIiHxh+34=",
    "y": "gB7BP7vo6rs9pmuZxavJhKTagJ7x7Ls/ej0wwtzah90="
  },
  "managed": null,
  "releasePolicy": null,
  "tags": null
}
```

Figure 41 : List ECC key

5.8.2 Importing on Windows

Use the Azure CLI to import the wrapped key to your Azure key vault, generated in the [previous steps](#).

1. Execute the following command to import an RSA tenant key:

>_ Console

```
> az keyvault key import --vault-name <keyvault> --name <WrappedKeyName> --byok-file "<wrappedkey>"
```

```
C:\Users\Utimaco>az keyvault key import --vault-name IG-testvault3 --name utimacotestKeyRSA3 --byok-file "Azure-BYOKRSAtest"
{
  "attributes": {
    "created": "2026-06-29T12:17:39+00:00",
    "enabled": true,
    "expires": null,
    "exportable": false,
    "hsmPlatform": "2",
    "notBefore": null,
    "recoverableDays": 90,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2026-06-29T12:17:39+00:00"
  },
  "key": {
    "crv": null,
    "d": null,
    "dp": null,
    "dq": null,
    "e": "AQAB",
    "k": null,
    "keyOps": [
      "encrypt",
      "decrypt",
      "sign",
      "verify",
      "wrapKey",
      "unwrapKey"
    ],
    "kid": "https://ig-testvault3.vault.azure.net/keys/utimacotestKeyRSA3/b1d1e91cce7443db8f66326cf997add2",
    "kty": "RSA-HSM",
    "n": "1v0zN2S1TGL1SE41nucf2Rh1HBRXS006Mo9AcuPwDT5g21wsMQnIzKcWuoZaijEzI14ItX7bCatMGaaZvx6VygWtXk9FKMY038oQK/v5J5EDT11qclZ0eE0oRiLPfXCCBistoVc/AFWx0VCp1K/90sHHAwWcJMeBwNFTJBdMMhmf307Ubp0rTCS19HsoHHZhdjbrT/IDmfS1s0ipMCDMeSpT421xkmD54Y+gN51fnV//OTpbjUvTBR00tqRf/jnwp02Z2CLxGehhipFTzk35ciGFTDCYe1n7vY1Hv8eagSSiDvADV1KPaZCY18EgrKTjmgxnmgLO2CubQ==",
    "p": null,
    "q": null,
    "qi": null,
  }
}
```

Figure 42 : Importing tenant key

Execute the following command to import an EC tenant key:

> _ Console

```
> az keyvault key import --vault-name <keyvault> --name <WrappedKeyName> --byok-file "<wrappedkey>" --kty EC --curve <curvename>
```

For **<curvename>** Azure supports P-256, P-384 and P-521. The curve name must match the actual key.

```
C:\Users\Utimaco>az keyvault key import --vault-name IG-testvault3 --name utimacotestKeyECC3 --byok-file "Azure-BYOKECCtest" --key EC --curve P-256
{
  "attributes": {
    "created": "2026-06-29T12:28:38+00:00",
    "enabled": true,
    "expires": null,
    "exportable": false,
    "hsmPlatform": "2",
    "notBefore": null,
    "recoverableDays": 90,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2026-06-29T12:28:38+00:00"
  },
  "key": {
    "crv": "P-256",
    "d": null,
    "dp": null,
    "dq": null,
    "e": null,
    "k": null,
    "keyOps": [
      "sign",
      "verify"
    ],
    "kid": "https://ig-testvault3.vault.azure.net/keys/utimacotestKeyECC3/9a25801f9cdf48ddb189a6767279da6c",
    "kty": "EC-HSM",
    "n": null,
    "p": null,
    "q": null,
    "qi": null,
    "t": null,
    "x": "VqtmVzBT+pALz0Oe51iHmhppurzSbm6neFFQLu1d8Gw=",
    "y": "wOiu+X/TXR+K2+rhbDkGXbxOoaJcII1V39YJdNJW28jU="
  },
  "managed": null,
  "releasePolicy": null,
  "tags": null
}
```

Figure 43 : Importing tenant key

+ Generate/Import ↻ Refresh ↶ Restore Backup 🔗 Manage deleted keys		
Name	Status	Expiration date
utimacotestKeyECC3	✓ Enabled	
utimacotestKeyRSA3	✓ Enabled	

Figure 44 : List importing tenant key

- Use the following command to check if the key has been successfully imported to the Azure Key vault:

> _ Console

```
> az keyvault key show --vault-name <keyvault> --name <WrappedKeyName>
```

You can also check if the key is visible on your Azure portal.

For the RSA Key

```

C:\Users\Utimaco>az keyvault key show --vault-name IG-testvault3 --name utimacotestKeyRSA3
{
  "attributes": {
    "created": "2026-06-29T12:17:39+00:00",
    "enabled": true,
    "expires": null,
    "exportable": false,
    "hsmPlatform": "2",
    "notBefore": null,
    "recoverableDays": 90,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2026-06-29T12:17:39+00:00"
  },
  "key": {
    "crv": null,
    "d": null,
    "dp": null,
    "dq": null,
    "e": "AQAB",
    "k": null,
    "keyOps": [
      "encrypt",
      "decrypt",
      "sign",
      "verify",
      "wrapKey",
      "unwrapKey"
    ],
    "kid": "https://ig-testvault3.vault.azure.net/keys/utimacotestKeyRSA3/b1d1e91cce7443db8f66326cf997add2",
    "kty": "RSA-HSM",
    "n": "1v0zN2S1TGL1SE41nucf2Rh1HBR5XSO06Mo9AcuPwDT5g21wsMQnIzKclwuozaiejEZI14ItX7bCatMGaaZvx6VygWTxk9FkMY030oQK/v5J5EDT11qcLZOeE00RiLPfXCCBIstoVc/AFWx0VCp1K/90sHHAMMcJM
BwNFTJBdMWhmf307Ubp0rTCS19HsoHHZhdjbrT/IDmf5IsOipMCDMe5pT421xkmD54Y+gM5ifnV//OTpbjUvTBR60tqRf/jnwp02ZZCLXGehhipFTzk3ScigFTDCYe1n7VY1Hv8eagSSIdVADV1KPaZC8Yi8REgrKTjmgxnr
gL02CubQ==",
    "p": null,
    "q": null,
    "qi": null,
    "t": null,
    "x": null
  }
}

```

Figure 45 : List RSA key

For the ECC Key

```
C:\Users\Utimaco>az keyvault key show --vault-name IG-testvault3 --name utimacotestKeyECC3
{
  "attributes": {
    "created": "2026-06-29T12:28:38+00:00",
    "enabled": true,
    "expires": null,
    "exportable": false,
    "hsmPlatform": "2",
    "notBefore": null,
    "recoverableDays": 90,
    "recoveryLevel": "Recoverable+Purgeable",
    "updated": "2026-06-29T12:28:38+00:00"
  },
  "key": {
    "crv": "P-256",
    "d": null,
    "dp": null,
    "dq": null,
    "e": null,
    "k": null,
    "keyOps": [
      "sign",
      "verify"
    ],
    "kid": "https://ig-testvault3.vault.azure.net/keys/utimacotestKeyECC3/9a25801f9cdf48ddb189a6767279da6c",
    "kty": "EC-HSM",
    "n": null,
    "p": null,
    "q": null,
    "qi": null,
    "t": null,
    "x": "VqtmVzBT+pALz0Oe51iHmhppurzSbm6neFFQLu1d8Gw=",
    "y": "wOiu+X/TXR+K2+rhbDkGXbxOoaJcI1V39YJdNJW28jU="
  },
  "managed": null,
  "releasePolicy": null,
  "tags": null
}
```

Figure 46 : List ECC key

5.9 FIPS mode

All the steps are identical to the above, when the HSM is in FIPS 140-2 approved mode. The only difference is that the backup of the entire key database is not possible. Please refer to additional documentation on the Utimaco product CD.

6 Troubleshooting

Error	Diagnosis
07.04.2023 08:20:01.771 [00011358:00011358] C_Login E: Error CKR_USER_PIN_NOT_INITIALIZED occurred. 07.04.2023 08:20:01.771 [00011358:00011358] C_Logout T: enter C_Logout(hSession: 0x00000001) 07.04.2023 08:20:01.771 [00011358:00011358] C_Logout T: leave C_Logout() 07.04.2023 08:20:01.771 [00011358:00011358] C_CloseSession T: enter C_CloseSession(hSession: 0x00000001) 07.04.2023 08:20:01.771 [00011358:00011358] C_CloseSession T: leave C_CloseSession() 07.04.2023 08:20:01.771 [00011358:00011358] C_Finalize T: enter C_Finalize(pReserved: 0) 07.04.2023 08:20:01.771 [00011358:00011358] C_Finalize T: leave C_Finalize()	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.

Table 6: List of errors and their diagnoses

7 Further Information

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

All u.trust GP HSM product documentation is also available at the Utimaco IS GmbH website:
<https://utimaco.com/>.

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 Str. 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 References

Reference	Title/Company	Document No.
[CSPKCS11RM]	CryptoServer PKCS#11 p11tool2 Reference Manual	2012-0004

Table 7: References

9.2 Command Summary

Command	Purpose
<code>mkdir -p /opt/utimaco/bin</code>	Create directory for Utimaco binaries
<code>mkdir -p /opt/utimaco/lib</code>	Create directory for Utimaco libraries
<code>cp ~/path_to_application_folder/lib/libcs_pkcs11_R3.so /opt/utimaco/lib</code>	Copy PKCS#11 library to target directory
<code>cp csadm p11tool2 /opt/utimaco/bin</code>	Copy administration tools to binary folder
<code>chmod +x /opt/utimaco/bin/csadm /opt/utimaco/bin/p11tool2</code>	Make tools executable
<code>mkdir /etc/utimaco</code>	Create configuration directory
<code>cp cs_pkcs11_R3.cfg /etc/utimaco</code>	Copy PKCS#11 configuration file

Command	Purpose
<pre>./p11tool2 slot=<slot_no> Label=<token_label> Login=ADMIN,ADMIN.key InitToken=<SO_PIN></pre>	Initialize token and create Security Officer
<pre>./p11tool2 slot=<slot_no> LoginSO=<SO_PIN> InitPin=<CryptoUser_PIN></pre>	Initialize user PIN for slot
<pre>rpm --import https:// packages.microsoft.com/keys/ microsoft.asc</pre>	Import Microsoft repository key
<pre>dnf install -y https:// packages.microsoft.com/config/ rhel/9.0/packages-microsoft- prod.rpm</pre>	Add Microsoft package repository
<pre>dnf install azure-cli</pre>	Install Azure CLI
<pre>az login</pre>	Authenticate to Azure account
<pre>az group create --location "<location>" --name "<resource_group>"</pre>	Create Azure resource group
<pre>az keyvault create --location "<location>" --name "<keyvault>" --resource-group "<resource_group>" --sku premium</pre>	Create Azure Key Vault (Premium)

Command	Purpose
<pre>az keyvault key create --name "<keyvault_key>" --vault-name "<keyvault>" --key RSA-HSM --size 2048 --ops import</pre>	Generate Key Exchange Key (KEK)
<pre>az keyvault key download --name "<keyvault_key>" --vault-name "<keyvault>" --file <keyvault_key>.publickey.pem</pre>	Download KEK public key
<pre>p11tool2 slot=<slot_no.> loginuser=<user_password> PubKeyAttr=CKA_LABEL="<tenant_public_key>",CKA_MODULUS_BITS=<keysize> PrvKeyAttr=CKA_LABEL="<tenant_private_key>",CKA_EXTRACTABLE=CK_TRUE GenerateKeyPair=RSA</pre>	Generate RSA tenant key in HSM
<pre>p11tool2 slot=<slot_no.> loginuser=<user_password> PubKeyAttr=CKA_LABEL="<tenant_public_key>",CKA_EC_PARAMS=oid:<curve_name> PrvKeyAttr=CKA_LABEL="<tenant_private_key>",CKA_EXTRACTABLE=CK_TRUE GenerateKeyPair=ECC</pre>	Generate ECC tenant key in HSM

Command	Purpose
<pre>byoktool Dev=<HSM_IP> LogonPass=<user,password> Label="<tenant_private_key>" CSP=azure PublicKey="<keyvaultkey>.pem" KID="<kid>" WrappedKey="<wrappedkey>"</pre>	Wrap tenant key for Azure import
<pre>az keyvault key import --vault- name <keyvault> --name <WrappedKeyName> --byok-file "wrappedkey"</pre>	Import wrapped RSA key into Azure Key Vault
<pre>az keyvault key import --vault- name <keyvault> --name <WrappedKeyName> --byok-file "wrappedkey" --key EC --curve <curvename></pre>	Import wrapped EC key into Azure Key Vault
<pre>az keyvault key show --vault-name <keyvault> --name <WrappedKeyName></pre>	Verify imported key in Azure

Table 8: Commands