

Integrating Oracle TDE using Azure Marketplace image with Azure Cloud HSM

(Oracle Database 12c Enterprise Edition)

Table of Contents

Summary	2
Prerequisites	2
System Requirements	2
Oracle TDE Installation Prerequisites	3
Installation and Configuration of Oracle TDE for Azure Cloud HSM	3
Create a VM using Azure Marketplace Image for Oracle DB.....	3
STEP 1: Create a New Virtual Machine to host your Oracle Database	3
STEP 2: Prepare the VM Environment.....	4
Create an Oracle Database	7
STEP 3: Create an Oracle Database	7
Automate Database Startup and Shutdown.....	9
STEP 4: Automate Oracle DB Startup and Shutdown	9

Install OpenSSL.....	11
STEP 5: Download and Install OpenSSL 1.1.1.1	11
Configure the Azure Cloud HSM Client Tools	14
STEP 6: Download and Install Azure Cloud HSM SDK.....	14
STEP 7: Configure the Azure Cloud HSM Client Tools	15
Configure System for TDE	16
STEP 8: System Prerequisites and System Setup.....	16
Appendix	24
Frequently Asked Questions	24

Summary

Azure Cloud HSM can be used to enhance the security of databases by providing dedicated hardware-based key management and cryptographic operations. It allows customers to encrypt their databases, ensuring that the data remains protected, even if the underlying infrastructure is compromised. Azure Cloud HSM supports Oracle TDE integration. With Oracle TDE, the database software encrypts data before storing it on disk. The data in the database is encrypted with a key. These keys are encrypted with the TDE master encryption key. Customers can store the TDE master encryption key in our Azure Cloud HSM, which provides additional security.

Prerequisites

System Requirements

- [Azure Cloud HSM Client SDK](#)
- [Azure Marketplace Oracle Database 12c Image](#)
- OpenSSL 1.1.1

- Azure Cloud HSM resource has been deployed, initialized, and configured.
- Azure Cloud HSM PKCS#11 library configured.

Important Note: Azure Cloud HSM supports Oracle TDE integration with Oracle database version 11 and 12. Our Azure Cloud HSM guide for Oracle TDE assumes you are using the Azure Marketplace image for Oracle Database 12c.

Oracle TDE Installation Prerequisites

- Copy of partition owner certificate “PO.crt” on application server.
- Known address of your HSM “hsm1.chsm-<resourcename>-<uniquestring>.privatelink.cloudhsm.azure.net”.
- Knowledge of Crypto User credentials
- Azure Marketplace Oracle Database VM Image

Installation and Configuration of Oracle TDE for Azure Cloud HSM

Create a VM using Azure Marketplace Image for Oracle DB

STEP 1: Create a New Virtual Machine to host your Oracle Database

You will need to create a new Virtual Machine in your existing client VNET. We recommend using Oracle Database 12C image from Azure Marketplace and following the steps documented below.

Reference: [Create an Oracle Database in a Azure Virtual Machine](#)

1. Use existing Cloud HSM Client Resource Group

Important Note: The Oracle TDE deployment in this example will be using the same VNET and private link that you created when following the Azure Cloud HSM Onboarding Guide. Access to the Azure Cloud HSM is through private endpoint. Failure to deploy Oracle into the same VNET and private link may result in access restrictions to the HSM from the Oracle DB.

2. Create Virtual Machine

```
az vm create --resource-group CHSM-CLIENT-RG --name oracledb12c --image Oracle:oracle-database:oracle_db_12_2_0_1_ee:12.2.01 --vnet-name chsmclient-vnet --subnet default --size Standard_DS2_v2 --admin-username oracleadmin --generate-ssh-keys --public-ip-sku Standard --location <RegionName>
```

Important Note: The **--vnet-name** and **--subnet** should be the same as during your initial private link setup, if you have not deployed your Azure Cloud HSM and configured private link, stop now, and refer to the [Azure Cloud HSM Onboarding Guide](#).

- **--generate-ssh-keys** will automatically create a key in the ~/.ssh/ directory, to use a specific set of keys, use the **--ssh-key-value** option.
- **Oracle:oracle-database:oracle_db_12_2_0_1_ee:12.2.01** is the image we're using from Azure Marketplace.

3. Create and attach a new disk for Oracle datafiles and FRA (flash recovery area). Size and SKU can differ depending on your needs.

```
az vm disk attach --name oradata01 --new --resource-group CHSM-CLIENT-RG --size-gb 64 --sku StandardSSD_LRS --vm-name oracledb12c
```

4. Open ports for connectivity. In this task you must configure some external endpoints for the database listener to use by setting up the Azure Network Security Group that protects your Virtual Machine.

- By default, the NSG name will take the format <VMName>NSG
- To open the endpoint that you use to access the Oracle database remotely, create a Network Security Group rule as follows:

```
az network nsg rule create --resource-group CHSM-CLIENT-RG --nsg-name oracledb12cNSG --name allow-oracle --protocol tcp --priority 1001 --destination-port-range 1521
```

- To open the endpoint that you use to access Oracle remotely, create a Network Security Group rule with az network nsg rule create as follows:

```
az network nsg rule create --resource-group CHSM-CLIENT-RG --nsg-name oracledb12cNSG --name allow-oracle-EM --protocol tcp --priority 1002 --destination-port-range 5502
```

STEP 2: Prepare the VM Environment

- 1. Connect to the Virtual Machine.** Replace publicIpAddress with the publicIpAddress value for your VM.

```
ssh -i <path to key> oracleadmin@<publicIpAddress>
```

- <path to key> should be /home/.ssh

2. Switch to the root user

```
sudo su -
```

3. Check for last created disk device that we will format for use holding Oracle datafiles

```
ls -alt /dev/sd* | head -1
```

- Expected output should return something like: [brw-rw----. 1 root disk 8, 32 Jul 11 18:32 /dev/sdc]

4. Format the device. As root user run parted on the device

- a. First create a disk label:

```
parted /dev/sdc mklabel gpt
```

- b. Then create a primary partition spanning the whole disk:

```
parted -a optimal /dev/sdc mkpart primary 0GB 64GB
```

- c. Finally check the device details by printing its metadata:

```
parted /dev/sdc print
```

- Expected output should return something like:

```
# parted /dev/sdc print
Model: Msft Virtual Disk (scsi)
Disk /dev/sdc: 68.7GB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:
Number Start End Size File system Name Flags
1 1049kB 64.0GB 64.0GB ext4 primary
```

5. Create a file system on the device partition.

```
mkfs -t ext4 /dev/sdc1
```

6. Create a mount point

```
mkdir /u02
```

7. Mount the disk.

```
mount /dev/sdc1 /u02
```

8. Change permissions on the mount point

```
chmod 777 /u02
```

9. Add the mount to the /etc/fstab file.

```
echo "/dev/sdc1      /u02      ext4 defaults    0 0" >> /etc/fstab
```

10. Update the /etc/hosts file with the public IP and hostname. Change the Public IP, VMName and RegionName to reflect your actual values.

```
echo "<Public IP> <VMName>.<RegionName>.cloudapp.azure.com <VMName>" >> /etc/hosts
```

11. Update the hostname file. Use the following command to add the domain name of the VM to the /etc/hostname file. This assumes you have created your resource group and VM in the same region:

```
sed -i 's/$/. <RegionName>\.cloudapp\.azure\.com &/' /etc/hostname
```

12. Open firewall ports. SELinux is enabled by default on the Marketplace image. We need to open the firewall to traffic for the database listening port 1521, and Enterprise Manager Express port 5502 by running the following commands as root user.

```
yum install firewalld
firewalld
firewall-cmd --zone=public --add-port=1521/tcp --permanent
firewall-cmd --zone=public --add-port=5502/tcp --permanent
firewall-cmd --reload
```

Create an Oracle Database

STEP 3: Create an Oracle Database

The Oracle software is already installed on the Marketplace image. Create an Oracle database as follows.

1. Switch to the oracle user:

```
sudo su - oracle
```

2. Start the database listener.

```
lsnrctl start
```

- Expected output should return something like:

```
LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 11-JUL-2023 18:56:39

Copyright (c) 1991, 2016, Oracle. All rights reserved.

Starting /u01/app/oracle/product/12.2.0/dbhome_1/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 12.2.0.1.0 - Production
Log messages written to /u01/app/oracle/diag/tnslsnr/oracledb12c/listener/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=oracledb12c.eastus.cloudapp.azure.com)(PORT=1521)))

Connecting to (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521))
STATUS of the LISTENER
-----
Alias                LISTENER
Version              TNSLSNR for Linux: Version 12.2.0.1.0 - Production
Start Date            11-JUL-2023 18:56:40
Uptime                0 days 0 hr. 0 min. 1 sec
Trace Level           off
Security              ON: Local OS Authentication
SNMP                  OFF
Listener Log File      /u01/app/oracle/diag/tnslsnr/oracledb12c/listener/alert/log.xml
```

Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=oracledb12c.eastus.cloudapp.azure.com)(PORT=1521)))

The listener supports no services

The command completed successfully

3. Create a data directory for the Oracle data files:

```
mkdir /u02/oradata
```

4. Run the Database Creation Assistant: Refer to Oracle documentation for the best database settings, the values used here are simply for testing configuration.

```
dbca -silent \ -createDatabase \ -templateName General_Purpose.dbc \ -gdbname oratest1 \ -sid oratest1 \ -responseFile NO_VALUE \ -characterSet AL32UTF8 \ -sysPassword OraPasswd1 \ -systemPassword OraPasswd1 \ -createAsContainerDatabase false \ -databaseType MULTIPURPOSE \ -automaticMemoryManagement false \ -storageType FS \ -datafileDestination "/u02/oradata/" \ -ignorePreReqs
```

- Expected output should return something like:

Prepare for db operation

10% complete

ing database files

40% complete

Creating and starting Oracle instance

42% complete

46% complete

50% complete

54% complete

60% complete

Completing Database Creation

66% complete

69% complete

70% complete

Executing Post Configuration Actions

100% complete

Database creation complete. For details check the logfiles at: /u01/app/oracle/cfgtoollogs/dbca/oratest1.

Database Information:

Global Database Name:oratest1

System Identifier(SID):oratest1

Look at the log file "/u01/app/oracle/cfgtoollogs/dbca/oratest1/oratest1.log" for further details.

5. Set Oracle Variables

- a. Before you connect, you need to set the environment variable ORACLE_SID:

```
export ORACLE_SID=oratest1
```

- b. You should also add the ORACLE_SID variable to the oracle users .rc file for future sign-ins using the following command:

```
echo "export ORACLE_SID=oratest1" >> ~oracle/.bashrc
```

6. Exit out as oracle user:

```
exit
```

Automate Database Startup and Shutdown

STEP 4: Automate Oracle DB Startup and Shutdown

The Oracle database by default doesn't automatically start when you restart the VM. To set up the Oracle database to start automatically, first sign in as root. Then, create and update system files.

1. Sign on as root.

```
sudo su -
```

Important Note: If you are logged in as oracle user and attempt to sign in as root some customers have reported issues with root password failing, try again. Recommend to 'exit' as an oracle user first then sign on as root.

2. Run the following command to change the automated startup flag from N to Y in the /etc/oratab file:

```
sed -i 's/:N/:Y/' /etc/oratab
```

3. Create a file named /etc/init.d/dbora and then paste and save the contents below:

```
vi /etc/init.d/dbora
```

```
#!/bin/sh
# chkconfig: 345 99 10
# Description: Oracle auto start-stop script.
#
# Set ORA_HOME to be equivalent to $ORACLE_HOME.
ORA_HOME=/u01/app/oracle/product/12.2.0/dbhome_1
ORA_OWNER=oracle

case "$1" in
'start')
    # Start the Oracle databases:
    # The following command assumes that the Oracle sign-in
    # will not prompt the user for any values.
    # Remove "&" if you don't want startup as a background process.
    su - $ORA_OWNER -c "$ORA_HOME/bin/dbstart $ORA_HOME" &
    touch /var/lock/subsys/dbora
    ;;

'stop')
    # Stop the Oracle databases:
    # The following command assumes that the Oracle sign-in
    # will not prompt the user for any values.
    su - $ORA_OWNER -c "$ORA_HOME/bin/dbshut $ORA_HOME" &
    rm -f /var/lock/subsys/dbora
    ;;
esac
```

4. Change permissions on files with chmod as follows:

```
chgrp dba /etc/init.d/dbora
chmod 750 /etc/init.d/dbora
```

5. Create symbolic links for startup and shutdown as follows:

```
In -s /etc/init.d/dbora /etc/rc.d/rc0.d/K01dbora  
In -s /etc/init.d/dbora /etc/rc.d/rc3.d/S99dbora  
In -s /etc/init.d/dbora /etc/rc.d/rc5.d/S99dbora
```

6. Exit out of Oracle Database

```
exit
```

7. To test your changes, restart the VM:

```
az vm restart -g CHSM-CLIENT-RG -n oracledb12c
```

Important Note: You can restart your Oracle VM from Azure Portal UI or from Azure CLI. For this example, we are restarting from CLI.

Install OpenSSL

STEP 5: Download and Install OpenSSL 1.1.1.1

The Azure Cloud HSM client SDK has a dependency on openssl 1.1.1, currently the distributed version for the Oracle Linux image available is openssl 1.0.2, this section will cover how to upgrade to 1.1.1t.

Reference: [How to Install the latest OpenSSL version from Source on Linux](#)

1. Connect to the Virtual Machine. Replace publicIpAddress with the publicIpAddress value for your VM.

```
ssh -i <path to key> oracleadmin@<publicIpAddress>
```

- <path to key> should be /home/.ssh

2. Sign on as root.

```
sudo su -
```

3. Install Tools

- a. As root, sudo su -, install the following packages.

```
yum group install 'Development Tools'
yum install -y perl-core zlib-devel wget vim
```

4. Download OpenSSL

- b. Go to the '/usr/local/src' directory and download the OpenSSL source code.

```
cd /usr/local/src/
wget https://www.openssl.org/source/openssl-1.1.1t.tar.gz --no-check-certificate
```

- c. Now extract the openssl.tar.gz file and go to the 'openssl' directory.

```
tar -xf openssl-1.1.1t.tar.gz
cd openssl-1.1.1t
```

5. Install OpenSSL

- d. Before installing the custom OpenSSL version to the system, let's check the installed version using the command below. We will **replace version 1.0.2** with version **OpenSSL 1.1.1**.

```
openssl version -a
```

- e. We will install the new OpenSSL version to the specific directory '/usr/local/ssl', and then enable the Link Libraries of OpenSSL, and configure the new binary PATH for OpenSSL.
 - i. Configure and compile OpenSSL with the commands below.

```
./config --prefix=/usr/local/ssl --openssldir=/usr/local/ssl shared zlib
make
```

- f. When the compile process is complete, install the OpenSSL using the command below. OpenSSL is installed in the '/usr/local/ssl' directory.

```
make install
```

6. Configure Link Libraries

- g. Next, we will configure the shared libraries for OpenSSL. The new OpenSSL binary will load library files from the '/usr/local/ssl/lib' directory. Go to the '/etc/ld.so.conf.d' directory and create new configuration file 'openssl-1.1.1.conf'.

```
cd /etc/ld.so.conf.d/
```

```
vim openssl-1.1.1t.conf
```

- h. Paste the openssl library path into the config file you just created.

```
/usr/local/ssl/lib
```

- i. Save and exit (in VIM, hit the esc key, then :w to save, followed by :q to exit)
- j. Reload the dynamic link, you should see the library path you just added along with the 1.1 libraries.

```
sudo ldconfig -v
```

7. Configure OpenSSL Binaries

- k. **Backup the current binaries**

```
mv /bin/openssl /bin/openssl.bak
```

- l. **Create a new file openssl.sh and paste in the below contents.**

```
vim /etc/profile.d/openssl.sh
```

```
#Set OPENSSL_PATH
OPENSSL_PATH="/usr/local/ssl/bin"
export OPENSSL_PATH
PATH=$PATH:$OPENSSL_PATH
export PATH
```

- m. **Make openssl.sh file executable.**

```
chmod +x /etc/profile.d/openssl.sh
```

- n. Load the OpenSSL environment and check the PATH bin directory using the commands below.

```
source /etc/profile.d/openssl.sh
echo $PATH
```

- o. **Now check the OpenSSL file, it should now be /usr/local/ssl/bin/openssl**

which openssl

8. Verify OpenSSL Version

```
openssl version -a
```

- Expected output should return something like:

Output should be OpenSSL 1.1.1t 7 Feb 2023

Configure the Azure Cloud HSM Client Tools

STEP 6: Download and Install Azure Cloud HSM SDK

This section assumes you have already successfully deployed an Azure Cloud HSM cluster and private endpoint but have NOT yet taken ownership of the HSM pool or configured the client tools. If you have already taken ownership of your HSM partition and want to test TDE, you can copy your existing PO.crt to your VM under the /opt/azurecloudhsm/cert directory and proceed as you would normally. The PO.crt must be present in the /cert directory.

For further instructions on how to configure client tools, activate your Cloud HSM and take ownership of your Cloud HSM please refer to the Azure Cloud HSM Onboarding Guide.

Customers can download the Azure Cloud HSM SDK and Client Tools from GitHub:

@ [microsoft/MicrosoftAzureCloudHSM](https://github.com/microsoft/MicrosoftAzureCloudHSM): Azure Cloud HSM SDK

- Linux Installation:** Customers can copy the Azure Cloud HSM deb or rpm package to their VM, then use dpkg to install it. The Azure Cloud HSM SDK is installed under /opt/azurecloudhsm. This is the directory for all Azure Cloud HSM management and client utilities including shared libraries for PKCS#11 and OpenSSL.

The example below demonstrates how to download and install the rpm package.

- `wget -p ~/ https://github.com/microsoft/MicrosoftAzureCloudHSM/releases/download/AzureCloudHSM-ClientSDK-2.0.1.2/AzureCloudHSM-ClientSDK-2.0.1.2-1.x86_64.rpm`
- `sudo rpm -i ./AzureCloudHSM-ClientSDK-2.0.1.2-1.x86_64.rpm --nodeps`

Important Note: As of the writing of this document, using the RPM installer for the SDK will result in an openssl dependency error. This error is limited to RHEL based OS earlier than 8, this Oracle Linux Server runs on 7.9. We have validated that the client works as expected despite the dependency error, so the current workaround is to install the RPM package using the --nodeps option.

Directory Example:

Azure Cloud HSM Management and Client Utilities

/opt/azurecloudhsm/bin

```
chsmVMAdmin@myAdminVM x + v
chsmVMAdmin@myAdminVM:/opt/azurecloudhsm/bin$ ls
azcloudhsm_application.cfg  azcloudhsm_mgmt_util  azcloudhsm_resource.cfg
azcloudhsm_client          azcloudhsm_mgmt_util.cfg  azcloudhsm_util
azcloudhsm_client.cfg      azcloudhsm_openssl_dynamic.conf  pkpspeed
```

Azure Cloud HSM Shared Libraries (PKCS#11 & OpenSSL)

/opt/azurecloudhsm/lib64

```
chsmVMAdmin@myAdminVM x + v
chsmVMAdmin@myAdminVM:/opt/azurecloudhsm/lib64$ ls
libazcloudhsm_api_socket.so  libazcloudhsm_openssl.so  libazcloudhsm_pkcs11.so
libazcloudhsm_api_socket.so.2  libazcloudhsm_openssl.so.2  libazcloudhsm_pkcs11.so.2
```

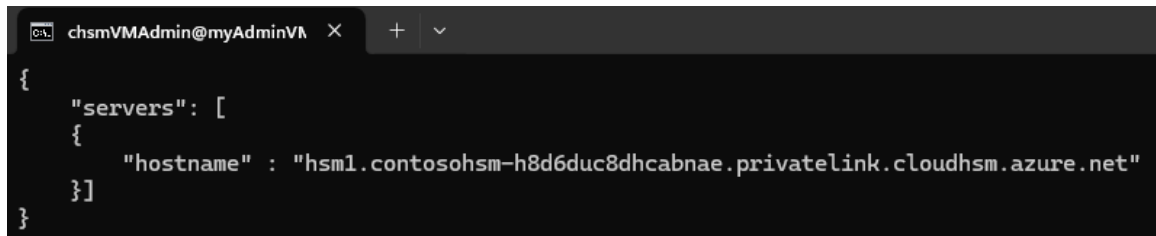
- b. **Ensure Azure Cloud HSM libraries are loaded.** The default library path is /opt/azurecloudhsm. Confirm this path and the libraries within it are present.

```
ldconfig -v
```

STEP 7: Configure the Azure Cloud HSM Client Tools

You will need to update the “hostname” property in the \bin\azcloudhsm_resource.cfg file. The azcloudhsm_resource.cfg is used to initialize the Azure Cloud HSM cluster. It must point to the private link FQDN for “hsm1” before the cluster is fully initialized as only hsm1 is currently running.

To find the FQDN that includes private link go to your resource group, and select Private Endpoint then select DNS Configuration. You will see configuration name and the FQDN that includes *.privatelink.cloudhsm.azure.net for each HSM instance. Use "hsm1" FQDN as the hostname for the \bin\azcloudhsm_resource.cfg file.



```
chsmVMAdmin@myAdminVM x + v
{
  "servers": [
    {
      "hostname" : "hsm1.contosohsm-h8d6duc8dhcabnae.privatelink.cloudhsm.azure.net"
    }
  ]
}
```

Configure System for TDE

STEP 8: System Prerequisites and System Setup

- Azure Cloud HSM PKCS#11 library (refer to the Azure Cloud HSM PKCS#11 Integration Guide).
- Azure Cloud HSM supports Oracle TDE integration with Oracle database versions 11 and 12. This guide assumes you are using the Azure Marketplace image for Oracle Database 12c.
- OpenSSL 1.1.1
- Azure Cloud HSM Cluster has been deployed, configured, initialized, and taken ownership of.

1. Set up your system.

- a. **Create a folder with the following name.** This is the Oracle TDE standard location.

```
sudo mkdir -p /opt/oracle/extapi/64/hsm
```

- b. **Copy the PKCS#11 library to the above path.**

```
sudo cp /opt/azurecloudhsm/lib64/libazcloudhsm_pkcs11.so /opt/oracle/extapi/64/hsm/
```

- c. **Set the permissions and ownership of the folder that contains the PKCS#11 library.**

```
sudo chown -R oracle:oinstall /opt/oracle
sudo chmod -R 775 /opt/oracle
```


- d. **Create a “wallet” directory, we will be using an HSM wallet but in testing I’ve found it doesn’t work properly unless this dir is created.**

```
sudo mkdir /u02/oradata/oratest1/wallet
```

Important Note: During testing we’ve found it doesn’t work properly unless the ‘wallet’ directory is created.

- e. **Paste the below line in \$ORACLE_HOME/network/admin/sqlnet.ora**

```
sudo su - oracle  
vim $ORACLE_HOME/network/admin/sqlnet.ora
```

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM)
```

- f. **Export azcloudhsm_daemon_id variable, paste the export line below in ~/.bashrc, then save and exit.**

```
vim ~/.bashrc
```

```
export azcloudhsm_daemon_id=1
```

- g. **Reboot the VM and run the following commands to confirm settings. We highly recommend not skipping this step, if any of these are not properly set the PKCS#11 library will fail**

```
sudo su - oracle  
echo $azcloudhsm_daemon_id  
echo $ORACLE_HOME  
echo $ORACLE_SID  
cat $ORACLE_HOME/network/admin/sqlnet.ora  
ls /opt/oracle/extapi/64/hsm
```

Important Note: You need to be logged in as oracle to see these properties above. If you are logged in as oracleadmin they will not show up when you try to validate these properties.

2. Validate TDE Integration

After confirming settings in the previous section, connect to the VM in three separate tabs. We will use one VM to run the Azure Cloud HSM client, one to run the Oracle database, and another as your Admin VM to do validation with the `azcloudhsm_util` found within our Azure Cloud HSM SDK.

Important Note: You may need to enter the following commands line by line as multiline pasting can cause issues in `sqlplus`

- a. **Connect to the database as an admin, but do not run any operations yet. If you have setup auto startup you do not need to run the “startup” command.**

```
sudo su oracle
sqlplus '/ as sysdba'
startup
```

Important Note: If you have setup auto startup you do not need to run the “startup” command.

- b. **In the 2nd VM tab, start the Azure Cloud HSM client.**

It's recommended for production to run the client daemon as a service. If you installed the Azure Cloud HSM SDK using deb or rpm, the client is not configured automatically to run as a service. The SDK during installation includes a service unit file under `/etc/systemd/system/azure-cloud-hsm.service`. To enable `azcloudhsm_client` to run as a service you will need to use the predefined `azure-cloud-hsm.service` file. You will then need to reload the Systemd configuration and enable the service to ensure its continuously running. For details on how to configure the client daemon to run as a service please reference the Azure Cloud HSM onboarding guide.

1. Open a terminal window and change directory to `/etc/systemd/system` where the Cloud HSM service unit file is located.

```
cd /etc/systemd/system
```

Example: `azure-cloud-hsm.service`

```
chsmVMAdmin@myAdminVM x + v
[Unit]
Description=Azure Cloud HSM Client Service
After=network.target

[Service]
Type=simple
ExecStart=/opt/azurecloudhsm/bin/azcloudhsm_client /opt/azurecloudhsm/bin/azcloudhsm_resource.cfg
ExecReload=/bin/kill -HUP $MAINPID
Restart=on-failure

# Logging
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=azure-cloud-hsm

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

2. Start and Enable the Cloud HSM Client service

```
sudo systemctl start azure-cloud-hsm.service
sudo systemctl enable azure-cloud-hsm.service
```

3. Check status of the Cloud HSM Client service

```
sudo systemctl status azure-cloud-hsm.service
```

```

chsmVMAdmin@myAdminVM:/etc/systemd/system$ sudo systemctl daemon-reload
chsmVMAdmin@myAdminVM:/etc/systemd/system$ sudo systemctl start azure-cloud-hsm.service
chsmVMAdmin@myAdminVM:/etc/systemd/system$ sudo systemctl enable azure-cloud-hsm.service
Created symlink /etc/systemd/system/multi-user.target.wants/azure-cloud-hsm.service → /etc/systemd/system/azure-cloud-hsm.service.
chsmVMAdmin@myAdminVM:/etc/systemd/system$ sudo systemctl status azure-cloud-hsm.service
● azure-cloud-hsm.service - Azure Cloud HSM Client Service
   Loaded: loaded (/etc/systemd/system/azure-cloud-hsm.service; enabled; vendor prese
   Active: active (running) since Tue 2024-08-27 19:08:11 UTC; 918ms ago
     Main PID: 11230 (azcloudhsm_clie)
        Tasks: 3 (limit: 4625)
       Memory: 892.0K
      CGroup: /system.slice/azure-cloud-hsm.service
              └─11230 /opt/azurecloudhsm/bin/azcloudhsm_client /opt/azurecloudhsm/bin/az

Aug 27 19:08:11 myAdminVM systemd[1]: Started Azure Cloud HSM Client Service.

```

4. Reload the Systemd configuration

```

sudo systemctl daemon-reload
sudo systemctl list-units --type=service --state=running

```

```

chsmVMAdmin@myAdminVM$ sudo systemctl list-units --type=service --state=running
UNIT                                LOAD    ACTIVE SUB    DESCRIPTION
accounts-daemon.service            loaded active running Accounts Service
atd.service                        loaded active running Deferred execution scheduler
azure-cloud-hsm.service             loaded active running Azure Cloud HSM Client Service
chrony.service                     loaded active running chrony, an NTP client/server

```

c. From the 2nd VM tab, start the azcloudhsm_util, login, and run the findKey operation, make note of the # of keys present. We will come back to this later.

```

cd /opt/azurecloudhsm/bin
./azcloudhsm_util
loginHSM -u CU -s <your username> -p <yourpassword>
findKey

```

[Example: loginHSM -u CU -s cu1 -p user1234]

d. Return to the window with the database running, we're now ready for testing. We will open a new hardware keystore, create a master encryption key, and then encrypt some tablespaces. Run the following sql commands.

- i. Use your crypto username and password to set the keystore to open

```
SQL> administer key management set keystore open identified by "<user>:<password>";
```

[Example: administer key management set keystore open identified by "cu1:user1234";]

- ii. This command generates the master encryption key, you should again see *keystore altered*. For a quick sanity check, you can return to the azcloudhsm_util window and run the findKey operation, you should have one more key than previously noted.

```
SQL> administer key management set key identified by "<user>:<password>";
```

[Example: administer key management set key identified by "cu1:user1234";]

- iii. This command is to validate that we have the wallet/keystore open, the "status" column should say OPEN. With the wallet open, we will now create an encrypted tablespace.

```
SQL> select * from v$encryption_wallet;
```

Important Note: If status shows CLOSED you may need to restart your Oracle DB to resolve the issue. If you received error code hardware security module failed with PKCS#11 error CKR_GENERAL_ERROR(5) check that you have liquid security client running.

- e. **Executing the following command will create an encrypted table space.**

```
SQL> CREATE TABLESPACE encrypted_ts  
DATAFILE '/u02/oradata/oratest1/encrypted_ts01.dbf' SIZE 128K  
AUTOEXTEND ON NEXT 64K  
ENCRYPTION USING 'AES256'  
DEFAULT STORAGE(ENCRYPT);  
.
```

- Expected output should return something like:

```
SQL> CREATE TABLESPACE encrypted_ts
DATAFILE '/u02/oradata/oratest1/encrypted_ts01.dbf' SIZE 128K
AUTOEXTEND ON NEXT 64K
ENCRYPTION USING 'AES256'
DEFAULT STORAGE(ENCRYPT);
.
  2    3    4    5
Tablespace created.
```

- f. Executing the following command will display the name, encryption algorithm, and encryption status of the tablespace we just created, the values should be as follows:

```
SQL> SELECT NAME, ENCRYPTIONALG, ENCRYPTEDTS FROM
V$ENCRYPTED_TABLESPACES, V$TABLESPACE
WHERE V$ENCRYPTED_TABLESPACES.TS# = V$TABLESPACE.TS#;
```

- Expected output should return something like:

```
SQL> SELECT NAME, ENCRYPTIONALG, ENCRYPTEDTS FROM
V$ENCRYPTED_TABLESPACES, V$TABLESPACE
WHERE V$ENCRYPTED_TABLESPACES.TS# = V$TABLESPACE.TS#;
  2    3
NAME                                ENCRYPT ENC
-----
ENCRYPTED_TS                        AES256  YES
```

Important Note: Encrypt == encryption algorithm. ENC == encryption status.

- g. Next, we'll create a table in the encrypted table space, and add some data to it and then retrieve that data

```
SQL> CREATE TABLE system.employee (first_name VARCHAR2(128),
last_name VARCHAR2(128),
empID NUMBER,
salary NUMBER(6))
TABLESPACE encrypted_ts;
```

```
SQL> INSERT INTO system.employee(first_name, last_name, empID, salary)
VALUES ('cavium', 'employee', 007, 100);
```

```
SQL> select * from system.employee;
```

- i. With the wallet open, this should return the values we just inserted into the table.

```
SQL> SELECT * FROM system.employee;
```

FIRST_NAME	

LAST_NAME	

EMPID	SALARY
-----	-----
cavium	
employee	
7	100

- h. Now we close the wallet and attempt to access this table data again. You should get the error “wallet is not open”. You have now validated master key creation and encrypted tablespace creation.

```
SQL> administer key management set keystore close identified by "<user>:<password>";
```

```
SQL> SELECT * FROM SYSTEM.EMPLOYEE;
```

[Example: administer key management set keystore close identified by "cu1:user1234";]

- Expected output should return something like:

```
SQL> administer key management set keystore close identified by
```

```
keystore altered.
```

```
SQL> SELECT * FROM system.employee
```

```
2 ;
```

```
SELECT * FROM system.employee
```

```
*
```

```
ERROR at line 1:
```

```
ORA-28365: wallet is not open
```

Appendix

For more information on TDE please visit [Transparent data encryption \(TDE\) | Microsoft Learn](#)

Frequently Asked Questions

- **Why do I get PKCS#11 errors when trying to set key or keystore?**

Ensure that you have the Azure Cloud HSM client running [sudo ./azcloudhsm_client azcloudhsm_resource.cfg]. An incorrect user, password or liquid security client not running or client failing to connect may result in the following error below.

Error Message:

ERROR at line 1:

ORA-28407: Hardware Security Module failed with PKCS#11 error

CKR_GENERAL_ERROR(5)

- **Why do I get a daemon socket connection error when running azcloudhsm_client or azcloudhsm_util?**

Ensure that you have a copy of the PO.crt under /cert directory and that you have configured the azcloudhsm_resource.cfg file to point to your Cloud HSM. This error message can also occur if you failed to deploy your Oracle DB using the same VNET and private link that you created following the Azure Cloud HSM Onboarding Guide. Access to the Azure Cloud HSM is through private endpoint. Failure to deploy Oracle into the same VNET and private link may result in access restrictions to the HSM from the Oracle DB.

Error Message:

Cfm3Initialize() returned app id : 01000000

session_handle 1000000

Cfm3GetPartitionInfo returned: 0x40000040

LIQUIDSECURITY: Daemon socket connection error