

# Integrating PKCS#11 with Azure Cloud HSM

## Table of Contents

|                                                                              |    |
|------------------------------------------------------------------------------|----|
| Summary .....                                                                | 2  |
| Prerequisites .....                                                          | 3  |
| System Requirements .....                                                    | 3  |
| PKCS#11 Install Prerequisites .....                                          | 3  |
| Azure Cloud HSM SDK Prerequisites.....                                       | 3  |
| STEP 1: Create a New Virtual Machine (Application Server) .....              | 3  |
| STEP 2: Download and Install Azure Cloud HSM SDK .....                       | 3  |
| STEP 3: Configure the Azure Cloud HSM Client Tools.....                      | 5  |
| STEP 4: Running azcloudhsm_client as a service .....                         | 6  |
| STEP 5: Verify crypto user exists .....                                      | 9  |
| How to configure and run your PKCS#11 application with Azure Cloud HSM ..... | 10 |
| Linux Installation .....                                                     | 10 |
| Windows Installation .....                                                   | 11 |
| Linux Example: Custom PKCS#11 Sample Application Template .....              | 12 |
| Windows Example: Custom PKCS#11 Sample Application Template.....             | 18 |
| Appendix .....                                                               | 25 |
| Frequently Asked Questions .....                                             | 25 |
| PKCS#11 Specifications .....                                                 | 31 |

|                                                      |    |
|------------------------------------------------------|----|
| Unsupported Key Types and Mechanisms.....            | 31 |
| Supported Key Types .....                            | 31 |
| Supported Mechanisms .....                           | 32 |
| Supported Key Generation Mechanisms .....            | 32 |
| Supported Key Encryption/Decryption Mechanisms ..... | 33 |
| Supported Sign/Verify Mechanisms .....               | 34 |
| Supported Key Hash/Digest Mechanisms.....            | 34 |
| Supported Key Wrap Mechanisms .....                  | 35 |
| Supported Key Derivation Mechanisms .....            | 35 |
| Supported API Operations .....                       | 35 |
| Supported Attributes .....                           | 37 |
| Support for Certificate Storage .....                | 41 |

## Summary

PKCS#11 serves as a widely recognized standard for executing cryptographic functions on hardware security modules (HSMs). Azure Cloud HSM provides compliant implementations of the PKCS#11 library, ensuring adherence to the PKCS#11 version 2.40 specifications. To use PKCS#11 with Azure Cloud HSM, the Azure Cloud HSM Client SDK and PKCS#11 library must be installed and configured accordingly.

# Prerequisites

## System Requirements

- Windows Server (2016, 2019, 2022) or Linux (Ubuntu 20.04, Ubuntu 22.04, RHEL 7, RHEL 8), CBL Mariner 2
- Azure Cloud HSM resource has been deployed, initialized, and configured.
- [Azure Cloud HSM Client SDK](#)

## PKCS#11 Install 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

*Important Note: Customers using any version of Windows Server should install the most recent version of the Visual C++ Redistributable.*

## Azure Cloud HSM SDK Prerequisites

### STEP 1: Create a New Virtual Machine (Application Server)

You will need to create a Virtual Machine in your client VNET. If you choose to use your existing Admin VM you can skip this step. This will serve as your Application VM to run your PKCS#11 application. If you choose to create a New VM for your application, then after provisioning the VM you will download the [Azure Cloud HSM SDK from GitHub](#) to this VM to configure your Cloud HSM to use PKCS#11. You can find details on how to deploy a Windows or Linux Virtual Machine using Azure CLI, PowerShell, ARM Template or Azure Portal below.

- [Create a Windows Virtual Machine](#)
- [Create a Linux Virtual Machine](#)

### STEP 2: Download and Install Azure Cloud HSM SDK

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

@ [microsoft/MicrosoftAzureCloudHSM: Azure Cloud HSM SDK](#)

- a. **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 deb package. Please note that rpm packages are also available to customers.

- `wget -p ~/ https://github.com/microsoft/MicrosoftAzureCloudHSM/releases/download/AzureCloudHSM-ClientSDK-2.0.1.2/AzureCloudHSM-ClientSDK-2.0.1.2.deb`
- `sudo dpkg --install ~/AzureCloudHSM-ClientSDK-2.0.1.2.deb`

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. **Windows Installation:** To install the Azure Cloud HSM SDK for Windows, download and run the Azure Cloud HSM SDK MSI (Windows Installer Package). This installer will automatically configure all necessary environment variables and dependencies and set up the Cloud HSM Client to run as a service. In the example below, running the Cloud HSM MSI deploys the SDK to C:\Program Files\Microsoft Azure Cloud HSM Client SDK

```
C:\Program Files\Microsoft Azure Cloud HSM Client SDK>dir /w
Volume in drive C is Windows
Volume Serial Number is C63C-E7A6

Directory of C:\Program Files\Microsoft Azure Cloud HSM Client SDK

[.]          [..]          [cert]        [client]      [libs]        [mgmt_util]
NOTICE.txt   [utils]        VERSION
              2 File(s)      4,127 bytes
              7 Dir(s)      121,962,545,152 bytes free
```

### Azure Cloud HSM Shared Library (PKCS#11)

\Microsoft Azure Cloud HSM Client SDK \libs\pkcs11

\Microsoft Azure Cloud HSM Client SDK \libs\pkcs11\include

## STEP 3: Configure the Azure Cloud HSM Client Tools

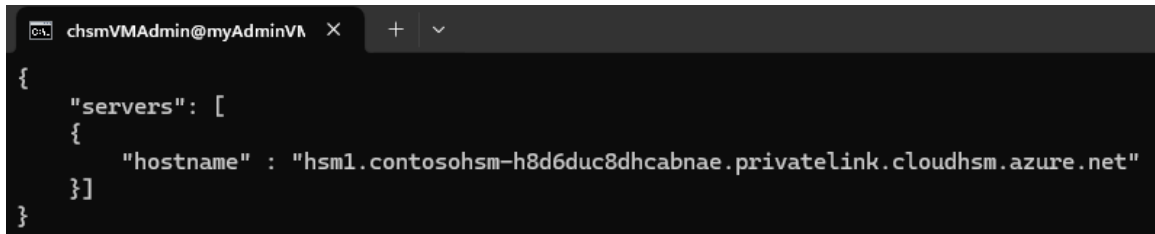
You will need to update the “hostname” property in the \bin\azcloudhsm\_resource.cfg file. The azcloudhsm\_resource.cfg is used to initialize the Azure Cloud HSM cluster. It must point to the private link FQDN for “hsm1” before the cluster is fully initialized as only hsm1 is currently running.

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.

| Configuration name             | FQDN                                                                | IP address |
|--------------------------------|---------------------------------------------------------------------|------------|
| privatelink-cloudhsm-azure-net |                                                                     |            |
| ✓                              | hsm1.chsm-mycloudhsm-fcdrgvhtfkxgcgg.privatelink.cloudhsm.azure.net | 10.0.2.4   |
| ✓                              | hsm2.chsm-mycloudhsm-fcdrgvhtfkxgcgg.privatelink.cloudhsm.azure.net | 10.0.2.5   |
| ✓                              | hsm3.chsm-mycloudhsm-fcdrgvhtfkxgcgg.privatelink.cloudhsm.azure.net | 10.0.2.6   |

**a. Linux Configuration:**

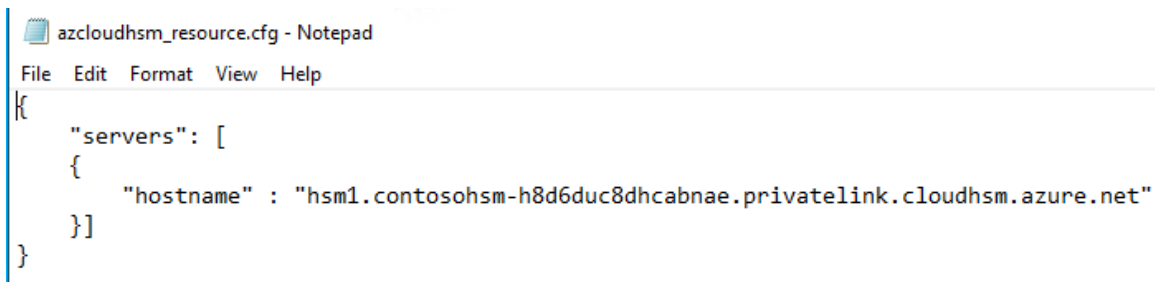
For Linux, the `azcloudhsm_resource.cfg` file to be updated is located under `/opt/azurecloudhsm/bin`



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

**b. Windows Configuration:**

For Windows, you will need to update `azcloudhsm_resource.cfg` in two locations, `\mgmt_util\azcloudhsm_resource.cfg` and `\client\azcloudhsm_resource.cfg`



```
azcloudhsm_resource.cfg - Notepad
File Edit Format View Help
{
  "servers": [
    {
      "hostname" : "hsm1.contosohsm-h8d6duc8dhcabnae.privatelink.cloudhsm.azure.net"
    }
  ]
}
```

## STEP 4: Running `azcloudhsm_client` as a service

We recommend running `azcloudhsm_client` as a service in a production environment since it needs to be continuously operational for the client-server model to establish connections and execute cryptographic operations. Manual execution of the `azcloudhsm_client` should be for development and testing purposes only.

**Linux (Recommended):**

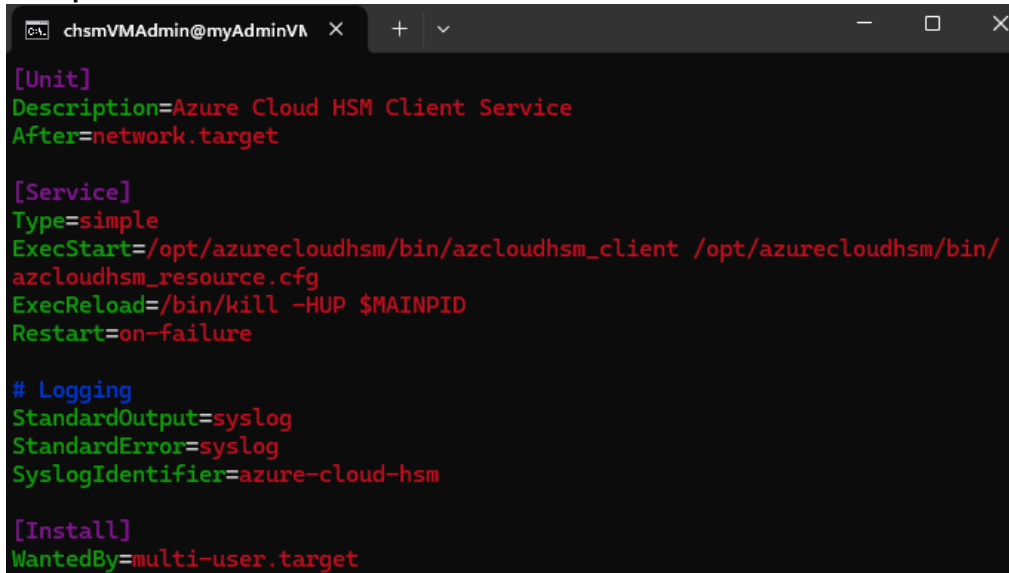
If you installed the Azure Cloud HSM SDK using `deb` or `rpm`, the client is not configured automatically to run as a service. The SDK during installation includes a service unit file under `/etc/systemd/system/azure-cloud-hsm.service`. To enable `azcloudhsm_client` to run as a service

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 by performing the following steps.

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**

A screenshot of a terminal window with a dark background. The window title is 'chsmVMAdmin@myAdminVM'. The terminal displays the contents of the 'azure-cloud-hsm.service' file in a syntax-highlighted format. The content includes sections for [Unit], [Service], # Logging, and [Install], with various configuration parameters like Description, After, Type, ExecStart, ExecReload, Restart, StandardOutput, StandardError, SyslogIdentifier, and WantedBy.

```
[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 x + - - - x
chsmVMAdmin@myAdminVM:/etc/systemd/system$ 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

```

#### Linux (Manual):

Start the client daemon if it is not running.

```
cd /opt/azurecloudhsm/bin
```

```
sudo ./azcloudhsm_client azcloudhsm_resource.cfg
```

You may also choose as an option with Linux to run the client daemon in the background using the following command.

```
sudo ./azcloudhsm_client azcloudhsm_resource.cfg > /dev/null 2>&1 &
```

#### Windows (Recommended):

If you installed the Azure Cloud HSM SDK using MSI, the client is already configured to run as a service. If the client is not running, you can open Services.msc, right-click on the Microsoft Azure Cloud HSM Client Service, and select "Start."



| Name                                                                                                                       | Description                                                         | Status  | Startup Type | Log On As    |
|----------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|---------|--------------|--------------|
|  Microsoft Azure Cloud HSM Client Service | Azure Cloud HSM Client Service connects applications with Cloud HSM | Running | Automatic    | Local System |

### Windows (Manual):

Start the client daemon if it is not running.

```
cd C:\Program Files\Microsoft Azure Cloud HSM Client SDK
.\azcloudhsm_client.exe .\azcloudhsm_resource.cfg
```

## STEP 5: Verify crypto user exists

A crypto user account is necessary to configure and run your PKCS#11 application with Azure Cloud HSM, as these applications operate under a crypto user.

- Use azcloudhsm\_util to login to your Azure Cloud HSM as Crypto User by executing the following commands below to validate CU can successfully login. If successful azcloudhsm\_util will return SUCCESS.

### Linux Installation:

```
./azcloudhsm_util
loginHSM -u CU -s cu1 -p user1234
```

### Windows Installation:

```
cd utils\azcloudhsm_util
.\azcloudhsm_util.exe
loginHSM -u CU -s cu1 -p user1234
```

- If you validated that you have a 'user' with Crypto User role successfully above, you can skip this step (c). If a 'user' with Crypto User roles does not exist, you will need to create a new user to run the PKCS#11 sample applications to validate your configuration. The example shows starting azcloudhsm\_mgmt\_util and administrator logging on as CO, then proceeds to create 'cu1' user with crypto user role.

### Linux Installation:

```
sudo ./azcloudhsm_mgmt_util ./azcloudhsm_resource.cfg
loginHSM CO admin adminpassword
createUser CU cu1 user1234
```

```
logoutHSM
loginHSM CU cu1 user1234
```

#### **Windows Installation:**

```
.\azcloudhsm_mgmt_util.exe .\azcloudhsm_resource.cfg
loginHSM CO admin adminpassword
createUser CU cu1 user1234
logoutHSM
loginHSM CU cu1 user1234
```

## How to configure and run your PKCS#11 application with Azure Cloud HSM

### Linux Installation

Below we created a PKCS#11 custom application example, then compile the program and generate the object file using gcc in the terminal. We then run the generated object file. While the PKCS#11 shared library exists under `/opt/azurecloudhsm/lib64` the .h files for Azure Cloud HSM PKCS#11 exist under `/opt/azurecloudhsm/pkcs11_headers/include`

You can install build essentials using your distribution package manager which is what we will use below to compile our PKCS#11 application.

```
sudo apt install build-essential
```

Copy the PKCS#11 sample application detailed further below and save as a .c file. In this example we saved the file as `cust_p11_app.c` under `/opt/azurecloudhsm`. You can then generate the object file by compiling the program using 'gcc'. The example below is using the custom PKCS#11 sample application template captured below under example.

```
sudo gcc -o cust_p11_app cust_p11_app.c -ldl
```

Ensure that you set the correct pin for your Azure Cloud HSM user credentials. Syntax is `username:password`. Your PKCS#11 application will use the pin parameter for the `C_Login` function.

```
char pPin[256] = "cu1:user1234";
```

Execute the custom PKCS#11 application from the example application template below.

```
sudo ./cust_p11_app -s cu1 -p user1234 -l /opt/azurecloudhsm/lib64/libazcloudhsm_pkcs11.so
```

## Windows Installation

Below we created a PKCS#11 custom application example, then compile the program and generate the object file using gcc in the terminal. We then run the generated object file. While the PKCS#11 shared library exists under \Microsoft Azure Cloud HSM Client SDK\libs the .h files for Azure Cloud HSM PKCS#11 exist under \Microsoft Azure Cloud HSM Client SDK\libs\pkcs11\include

1. Open PowerShell prompt as 'Administrator'
2. Run the following command to Set Execution Policy and Download Chocolatey:

```
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol =  
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object  
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

3. Run the following command for MinGW-w64 Installation:

```
choco install mingw
```

Copy the PKCS#11 sample application detailed further below and save as a .c file. In this example we saved the file as cust\_p11\_app.c under \Microsoft Azure Cloud HSM Client SDK. You can then generate the object file by compiling the program using 'gcc'. The example below is using the custom PKCS#11 sample application template captured below under example.

```
gcc -o cust_p11_app.exe cust_p11_app.c
```

Ensure that you set the correct pin for your Azure Cloud HSM user credentials. Syntax is username:password. Your PKCS#11 application will use the pin parameter for the C\_Login function.

```
char pPin[256] = "cu1:user1234";
```

Execute the custom PKCS#11 application from the example application template below.

```
.\cust_p11_app.exe -s cu1 -p user1234 -i "C:\Program Files\Microsoft Azure Cloud HSM Client SDK\libs\pkcs11\azcloudhsm_pkcs11.dll"
```

**Important Note:** We've provided two PKCS#11 applications samples, one for Linux customers and the other for Windows customers. The Linux samples makes use of `dlopen` (`dlopen`) to load the PKCS#11 library while Windows makes use of `LoadLibraryA` (`LoadLibraryA`) to load the PKCS#11 library. The `dlopen` is a set of functions that allows runtime dynamic library loading. It is standardized in the POSIX. Windows also provide similar routines, but not in a POSIX-compatible way which is why for Windows we've provided a sample using `LoadLibraryA` (`LoadLibraryA`) which is a standard Windows function.

**Important Note:** For your PKCS#11 application to execute successfully the `azcloudhsm_application.cfg` file must exist in the same location as your PKCS#11 application and the `azcloudhsm_client` must be running. If the `azcloudhsm_client` is not running, you will receive a failed socket

*connection. If the azcloudhsm\_client is not running your application will fail and you will not be able to initialize, create sessions, login or perform any Azure Cloud HSM or PKCS#11 functions.*

## Linux Example: Custom PKCS#11 Sample Application Template

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include <ctype.h>
#include <getopt.h>
#include "dlfcn.h"
#include "pkcs11_headers/include/cryptoki.h"
#include "pkcs11_headers/include/pkcs11t.h"
```

```
CK_FUNCTION_LIST_PTR func_list = NULL;
static void *module = NULL;
```

```
#define CAVIUM_SLOT 1
unsigned long int Private = 0;
unsigned long int Public = 0;
```

```
/* This section contains the standard .h files and user credentials. "cu1:user1234" represents the username and password that are initialized to log in to Azure Cloud HSM. */
```

```
char pPin[256] = "cu1:user1234";
```

```
/* This function loads the library; here dlopen is used. */
```

```
static int inline load_lib(const char *libpath)
```

```
{
    CK_RV rv = -1;
    CK_C_GetFunctionList get_function_list;
```

```
/* loads the PKCS#11 library present in libpath */
```

```

printf("[INFO] Azure Cloud HSM - Loading PKCS#11 library.\n");
module = dlopen(libpath, RTLD_NOW);

if (!module) {
    printf("couldn't open library: %s: %s", libpath, dlerror());
    return -1;
}

/* Look up the appropriate function in library */
printf("[INFO] Azure Cloud HSM - C_GetFunctionList\n");
get_function_list = (CK_C_GetFunctionList) dlsym(module, "C_GetFunctionList");
if (!get_function_list) {
    printf("couldn't find function 'C_GetFunctionList' in library: %s: %s", libpath, dlerror());
    return -1;
}

/* Makes all PKCS#11 functions available in the form of a function list by returning func_list */
rv = (get_function_list) (&func_list);
if (rv != CKR_OK || !func_list) {
    printf("couldn't get function list: %d", (int) rv);
    return -1;
}
return rv;
}

/* This function closes the object opened by the previous application. */
static int inline lib_close()
{
    if (module)
        dlclose(module);
    return 0;
}

/* Main function */

```

```

CK_RV main_func()
{
    CK_SESSION_HANDLE session_rw = 0;
    CK_SLOT_ID slot = CAVIUM_SLOT;
    CK_FUNCTION_LIST_PTR list;
    CK_INFO info = { };
    CK_TOKEN_INFO token_info = { };
    CK_RV rv = CKR_OK;
    int n_pin = strlen(pPin);
    int s_done = 0;

    rv = (func_list->C_GetFunctionList) (&list);
    if (CKR_OK != rv) {
        printf("\nCall through function list %08lx\n", rv);
        goto clean_up;
    }

    if (memcmp(list, func_list, sizeof(*list)) != 0)
        printf("\nCall doesn't return same data as library C_GetFunctionList entry point\n");

    printf("[INFO] Azure Cloud HSM – C_Initialize\n");
    rv = (func_list->C_Initialize) (NULL);
    if (CKR_OK != rv) {
        printf("\nC_Initialize() failed with %08lx\n", rv);
        goto clean_up;
    }

    printf("[INFO] Azure Cloud HSM – C_GetInfo\n");
    rv = (func_list->C_GetInfo) (&info);
    if (CKR_OK != rv) {
        printf("\nCall through function list %08lx\n", rv);
        goto clean_up;
    }
}

```

```

printf("[INFO] Azure Cloud HSM – Retrieve access token, C_GetTokenInfo\n");
rv = (func_list->C_GetTokenInfo) (slot, &token_info);
if (CKR_OK != rv) {
    printf("\nCall through function list %08lx\n", rv);
    goto clean_up;
}

printf("[INFO] Azure Cloud HSM – Start session with specified token, C_OpenSession\n");
rv = (func_list->C_OpenSession) (slot, CKF_SERIAL_SESSION | CKF_RW_SESSION, NULL, NULL, &session_rw);
if (CKR_OK != rv) {
    printf("\nC_OpenSession() failed with %08lx\n", rv);
    goto clean_up;
}
s_done = 1;

printf("[INFO] Azure Cloud HSM – Login with PIN, C_Login\n");
rv = (func_list->C_Login) (session_rw, CKU_USER, (CK_UTF8CHAR_PTR) pPin, n_pin);
if (CKR_OK != rv) {
    printf("[ERROR] Failed C_Login\n");
    goto clean_up;
}

/*****[INSERT CUSTOM P11 APP CODE HERE]*****/

printf("Add Your PKCS#11 Code Here\n");

/*****[END OF CUSTOM P11 APP CODE]*****/

/* Clean Up */
clean_up:
#ifdef USE_FIND_KEY
    if (s_done) {
        if (Private)
            (func_list->C_DestroyObject) (session_rw, Private);
    }
}

```

```

        if (Public)
            (func_list->C_DestroyObject) (session_rw, Public);
        Public = Private = 0;
    }
#endif

if (s_done) {
    /* logout */
    rv = func_list->C_Logout(session_rw);
    if (CKR_OK != rv) {
        printf("\nC_Logout() failed with %08lx\n", rv);
    }

    /* close session */
    rv = func_list->C_CloseSession(session_rw);
    if (CKR_OK != rv) {
        printf("\nC_CloseSession() failed with %08lx\n", rv);
    }
}

/* Application Shutdown */
rv = func_list->C_Finalize(NULL);
if (CKR_OK != rv) {
    printf("\nC_Finalize() failed with %08lx\n", rv);
}
return rv;
}

int main(int argc, char *argv[])
{
    CK_RV rv = 0;
    char *username = NULL;
    char *password = NULL;
    char *libname = NULL;

```



```
char buName = FALSE;
char bPassword = FALSE;
char bLibray = FALSE;
int cmd_options = 0;

if (argc < 7) {
    printf("\nMissing arguments: The format is %s -s <username> -p <password> -l <pkcs-lib> \n\n", argv[0]);
    return 0;
}

if (argc > 7) {
    printf("\nToo many arguments: The format is %s -s <username> -p <password> -l <pkcs-lib>\n\n", argv[0]);
    return 0;
}

while ((cmd_options = getopt(argc, argv,"s:p:l:")) != -1) {
    switch (cmd_options) {
        case 's' : username = optarg;
            buName = TRUE;
            break;
        case 'p' : password = optarg;
            bPassword = TRUE;
            break;
        case 'l' : libname = optarg;
            bLibray = TRUE;
            break;
        default: printf("\nWrong arguments: The format is %s -s <username> -p <password> -l <pkcs-lib>\n\n",argv[0]);
            return 0;
    }
}

if (!buName || !bPassword || !bLibray) {
    printf("\nWrong arguments: The format is %s -s <username> -p <password> -l <pkcs-lib>> \n\n", argv[0]);
    return 0;
}
```

```

}

if (load_lib(libname)) {
    printf("\n Error: loading library %s\n ", libname);
    rv = 0;
    goto end;
}

if (!module) {
    printf("\n Error: module loading failure %s\n ", libname);
    rv = 0;
    goto end;
}

/* prepare the pPin with given username and password */
printf("[INFO] Azure Cloud HSM – Preparing PIN with given username and password.\n");
snprintf(pPin, sizeof(pPin), "%s:%s", username, password);

rv = main_func();
if (rv != CKR_OK) {
    printf("Error while running the tests\n");
}

end:
    lib_close();
    exit(0);
}

```

## Windows Example: Custom PKCS#11 Sample Application Template

```

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

```

```
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include <ctype.h>
#include <getopt.h>
#include "libs\pkcs11\include\cryptoki.h"
#include "libs\pkcs11\include\pkcs11t.h"
```

```
CK_FUNCTION_LIST_PTR func_list = NULL;
static void *module = NULL;
```

```
#define CAVIUM_SLOT 1
unsigned long int Private = 0;
unsigned long int Public = 0;
```

```
/* This section contains the standard .h files and user credentials. "cu1:user1234" represents the username and password that are initialized to log in to Azure Cloud HSM. */
```

```
char pPin[256] = "cu1:user1234";
```

```
/* This function loads the library; here LoadLibraryA is used. */
```

```
static int inline load_lib(const char *libpath)
```

```
{
    CK_RV rv = -1;
    CK_C_GetFunctionList get_function_list;
```

```
/* loads the PKCS#11 library present in libpath */
```

```
printf("[INFO] Azure Cloud HSM - Loading PKCS#11 library.\n");
module = LoadLibraryA(libpath);
```

```
if (!module) {
    printf("couldn't open library");
    return -1;
}
```

```

/* Look up the appropriate function in library */
printf("[INFO] Azure Cloud HSM – C_GetFunctionList\n");
get_function_list = (CK_C_GetFunctionList)GetProcAddress(module,"C_GetFunctionList");
if (!get_function_list) {
    printf("couldn't find function 'C_GetFunctionList' in library");
    return -1;
}

/* Makes all PKCS#11 functions available in the form of a function list by returning func_list */
rv = (get_function_list) (&func_list);
if (rv != CKR_OK || !func_list) {
    printf("couldn't get function list: %d", (int) rv);
    return -1;
}
return rv;
}

/* This function closes the object opened by the previous application. */
static int inline lib_close()
{
    if (module)
        FreeLibrary(module);
    return 0;
}

/* Main function */
CK_RV main_func()
{
    CK_SESSION_HANDLE session_rw = 0;
    CK_SLOT_ID slot = CAVIUM_SLOT;
    CK_FUNCTION_LIST_PTR list;
    CK_INFO info = { };
    CK_TOKEN_INFO token_info = { };

```

```
CK_RV rv = CKR_OK;
int n_pin = strlen(pPin);
int s_done = 0;

rv = (func_list->C_GetFunctionList) (&list);
if (CKR_OK != rv) {
    printf("\nCall through function list %08lx\n", rv);
    goto clean_up;
}

if (memcmp(list, func_list, sizeof(*list)) != 0)
    printf("\nCall doesn't return same data as library C_GetFunctionList entry point\n");

printf("[INFO] Azure Cloud HSM – C_Initialize\n");
rv = (func_list->C_Initialize) (NULL);
if (CKR_OK != rv) {
    printf("\nC_Initialize() failed with %08lx\n", rv);
    goto clean_up;
}

printf("[INFO] Azure Cloud HSM – C_GetInfo\n");
rv = (func_list->C_GetInfo) (&info);
if (CKR_OK != rv) {
    printf("\nCall through function list %08lx\n", rv);
    goto clean_up;
}

printf("[INFO] Azure Cloud HSM – Retrieve access token, C_GetTokenInfo\n");
rv = (func_list->C_GetTokenInfo) (slot, &token_info);
if (CKR_OK != rv) {
    printf("\nCall through function list %08lx\n", rv);
    goto clean_up;
}
```

```

printf("[INFO] Azure Cloud HSM – Start session with specified token, C_OpenSession\n");
rv = (func_list->C_OpenSession) (slot, CKF_SERIAL_SESSION | CKF_RW_SESSION, NULL, NULL, &session_rw);
if (CKR_OK != rv) {
    printf("\nC_OpenSession() failed with %08lx\n", rv);
    goto clean_up;
}
s_done = 1;

printf("[INFO] Azure Cloud HSM – Login with PIN, C_Login\n");
rv = (func_list->C_Login) (session_rw, CKU_USER, (CK_UTF8CHAR_PTR) pPin, n_pin);
if (CKR_OK != rv) {
    printf("[ERROR] Failed C_Login\n");
    goto clean_up;
}

/*****[INSERT CUSTOM P11 APP CODE HERE]*****/

printf("Add Your PKCS#11 Code Here\n");

/*****[END OF CUSTOM P11 APP CODE]*****/

/* Clean Up */
clean_up:
#ifdef USE_FIND_KEY
    if (s_done) {
        if (Private)
            (func_list->C_DestroyObject) (session_rw, Private);
        if (Public)
            (func_list->C_DestroyObject) (session_rw, Public);
        Public = Private = 0;
    }
#endif

if (s_done) {

```

```

/* logout */
rv = func_list->C_Logout(session_rw);
if (CKR_OK != rv) {
    printf("\nC_Logout() failed with %08lx\n", rv);
}

/* close session */
rv = func_list->C_CloseSession(session_rw);
if (CKR_OK != rv) {
    printf("\nC_CloseSession() failed with %08lx\n", rv);
}
}

/* Application Shutdown */
rv = func_list->C_Finalize(NULL);
if (CKR_OK != rv) {
    printf("\nC_Finalize() failed with %08lx\n", rv);
}
return rv;
}

int main(int argc, char *argv[])
{
    CK_RV rv = 0;
    char *username = NULL;
    char *password = NULL;
    char *libname = NULL;
    char buName = FALSE;
    char bPassword = FALSE;
    char bLibray = FALSE;
    int cmd_options = 0;

    if (argc < 7) {
        printf("\nMissing arguments: The format is %s -s <username> -p <password> -l <pkcs-lib> \n\n", argv[0]);
    }

```

```

    return 0;
}

if (argc > 7) {
    printf("\nToo many arguments: The format is %s -s <username> -p <password> -l <pkcs-lib>\n\n", argv[0]);
    return 0;
}

while ((cmd_options = getopt(argc, argv, "s:p:l:")) != -1) {
    switch (cmd_options) {
        case 's' : username = optarg;
            buName = TRUE;
            break;
        case 'p' : password = optarg;
            bPassword = TRUE;
            break;
        case 'l' : libname = optarg;
            bLibray = TRUE;
            break;
        default: printf("\nWrong arguments: The format is %s -s <username> -p <password> -l <pkcs-lib>\n\n", argv[0]);
            return 0;
    }
}

if (!buName || !bPassword || !bLibray) {
    printf("\nWrong arguments: The format is %s -s <username> -p <password> -l <pkcs-lib>> \n\n", argv[0]);
    return 0;
}

if (load_lib(libname)) {
    printf("\n Error: loading library %s\n ", libname);
    rv = 0;
    goto end;
}

```



```

if (!module) {
    printf("\n Error: module loading failure %s\n ", libname);
    rv = 0;
    goto end;
}

/* prepare the pPin with given username and password */
printf("[INFO] Azure Cloud HSM – Preparing PIN with given username and password.\n");
snprintf(pPin, sizeof(pPin), "%s:%s", username, password);

rv = main_func();
if (rv != CKR_OK) {
    printf("Error while running the tests\n");
}

end:
    lib_close();
    exit(0);
}

```

## Appendix

### Frequently Asked Questions

- **C\_Login failed with error code: 0xa3.**
  - Encountering (0xa3) indicates a login failure that the issue might be related to the format of the PIN parameter you're passing or incorrect PIN (password).

- The PIN should be provided in the following format: `char pPin[256] = "crypto_user:user123"`; The pPin value should follow the structure of "`<username>:<password>`". Please verify that you are providing the pPin in this format, as any deviation might result in a login failure.
- **C\_Initialize() failed with 00000005 (Failed to connect socket).**
  - Encountering a failed socket connection during C\_Initialize might be related to the `azcloudhsm_client` not running on your system. For your PKCS#11 applications to execute successfully the `azcloudhsm_client` must be running. You can start the `azcloudhsm_client` by ensuring its running as a service or by manually running the following command in a separate terminal.

**Linux (Recommended):**

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 by performing the following steps.

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

```

#### Linux (Manual):

Start the client daemon if it is not running.

```
cd /opt/azurecloudhsm/bin
```

```
sudo ./azcloudhsm_client azcloudhsm_resource.cfg
```

You may also choose as an option with Linux to run the client daemon in the background using the following command.

```
sudo ./azcloudhsm_client azcloudhsm_resource.cfg > /dev/null 2>&1 &
```

#### Windows (Recommended):

If you installed the Azure Cloud HSM SDK using MSI, the client is already configured to run as a service. If the client is not running, you can open Services.msc, right-click on the Microsoft Azure Cloud HSM Client Service, and select "Start."

| Name                                                                                                                       | Description                                                         | Status  | Startup Type | Log On As    |
|----------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|---------|--------------|--------------|
|  Microsoft Azure Cloud HSM Client Service | Azure Cloud HSM Client Service connects applications with Cloud HSM | Running | Automatic    | Local System |

### Windows (Manual):

Start the client daemon if it is not running.

```
cd C:\Program Files\Microsoft Azure Cloud HSM Client SDK
.\azcloudhsm_client.exe .\azcloudhsm_resource.cfg
```

- **azcloudhsm\_application.cfg is not present (Cannot find file).**
  - The file location of the azcloudhsm\_application.cfg is different for Linux and Windows. You may need to copy the azcloudhsm\_application.cfg file from the default location below to the location of the PKCS#11 application you are running. They should both exist in the same directly.
    - **Linux File Location:** /opt/azurecloudhsm/bin
    - **Windows File Location:** C:\Program Files\Microsoft Azure Cloud HSM Client SDK\utils\azcloudhsm\_util\azcloudhsm\_application.cfg
- **How does the PKCS#11 library know how to find the client configuration for Linux.**
  - The PKCS#11 library knows how to find the client configuration as you must have a copy of your partition owner certificate "PO.crt" on the application server that is running your application and using the PKCS#11 library. In addition to PO certificate you have to update azcloudhsm\_resource.cfg on your application server that has the SDK installed to point to your Azure Cloud HSM (i.e hsm1.chsm-<resourcename>-<uniquestring>.privatelink.cloudhsm.azure.net). The azcloudhsm\_client must be running on your application server which connects to your Azure Cloud HSM. Finally, you must specify a PIN within your PKCS#11 application using syntax <username>:<password> which is used when calling C\_Login to your Azure Cloud HSM.
  - You will need to #include pkcs11\_headers/include/cryptoki.h and pkcs11\_headers/include/pkcs11t.h in your PKCS#11 application to use the Azure Cloud HSM PKCS#11 library.
- **How does the PKCS#11 library (azcloudhsm\_pkcs11.dll) for Windows know how to find the client configuration?**
  - The azcloudhsm\_pkcs11.dll under Azure Cloud HSM Windows SDK knows how to find the client configuration as you must have a copy of your partition owner certificate "PO.crt" on the application server that is running your application and using the PKCS#11 library. In addition to PO certificate you have to update azcloudhsm\_resource.cfg on your application server that has the SDK installed to point to your Azure Cloud HSM (i.e hsm1.chsm-<resourcename>-<uniquestring>.privatelink.cloudhsm.azure.net). The azcloudhsm\_client must run on your application server which connects to

your Azure Cloud HSM. Finally, you must specify a PIN within your PKCS#11 application using syntax <username>:<password> which is used when calling C\_Login to your Azure Cloud HSM.

- You will need to #include pkcs11\_headers\include\cryptoki.h and pkcs11\_headers\include\pkcs11t.h in your PKCS#11 application to use the Azure Cloud HSM PKCS#11 library.

- **How does the PKCS#11 library (azcloudhsm\_pkcs11.dll) integrate with OpenSC (Open-Source Smart Card Tools)?**

Download OpenSC MSI for Windows: [OpenSC: Open source smart card tools and middleware. PKCS#11/MiniDriver/Tokend \(github.com\)](https://github.com/OpenSC/OpenSC/wiki/MiniDriver/Tokend)

- Customers must use the "--sensitive" and "--private" arguments when using OpenSC tools as the sensitive and private attributes are not allowed to be 0. Without using those arguments, the sensitive and private attributes are set to 0 by OpenSC tooling and will cause an exception as Azure Cloud HSM always imposes sensitive and private behavior.
  - *AES Wrapping Example: pkcs11-tool.exe --sensitive --private --module "C:\Program Files\Microsoft Azure Cloud HSM Client SDK\libs\pkcs11\azcloudhsm\_pkcs11.dll" --login --login-type user --pin cu1:user1234 --keygen --key-type AES:32*
  - *ECC P521 Example: pkcs11-tool.exe --sensitive --private --module "C:\Program Files\Microsoft Azure Cloud HSM Client SDK\libs\pkcs11\azcloudhsm\_pkcs11.dll" --login --login-type user --pin cu1:user1234 --keypairgen --key-type EC:prime256v1 --usage-sign*

```
Administrator: Command Prompt

C:\Program Files\OpenSC Project\OpenSC\tools>pkcs11-tool.exe --sensitive --private --module "C:\Program Files\Microsoft
Azure Cloud HSM Client SDK\libs\pkcs11\azcloudhsm_pkcs11.dll" --login --login-type user --pin cu1:user1234 --keygen --ke
y-type AES:32
azcloudhsm_application.cfg is not present
azcloudhsm_application.cfg is not present
Using slot 0 with a present token (0x1)
Key generated:
Secret Key Object; AES length 32
warning: PKCS11 function C_GetAttributeValue(VALUE) failed: rv = CKR_ATTRIBUTE_SENSITIVE (0x11)

    label:
    Usage:      encrypt, decrypt, sign, verifywarning: PKCS11 function C_GetAttributeValue(VERIFY_RECOVER) failed: rv = CK
R_ATTRIBUTE_TYPE_INVALID (0x12)

    Access:      sensitive, always sensitive, never extractable, local

C:\Program Files\OpenSC Project\OpenSC\tools>pkcs11-tool.exe --sensitive --private --module "C:\Program Files\Microsoft
Azure Cloud HSM Client SDK\libs\pkcs11\azcloudhsm_pkcs11.dll" --login --login-type user --pin cu1:user1234 --keypairgen
--key-type EC:prime256v1 --usage-sign
azcloudhsm_application.cfg is not present
azcloudhsm_application.cfg is not present
Using slot 0 with a present token (0x1)
Key pair generated:
Private Key Object; EC
    label:
    Usage:      sign
    Access:      sensitive, always sensitive, extractable, local
Public Key Object; EC EC_POINT 256 bits
    EC_POINT:    044104f713778bb72e494303ee3f47e11c00d1e92442ba99112df0d0395d666c4f4c1e7ea0f057a0d7ec0929c67dfbb9cb117d4f69
5d7a4f1dde95fa0bc56514ee9156
    EC_PARAMS:    06082a8648ce3d030107 (OID 1.2.840.10045.3.1.7)
    label:
    Usage:      verify
    Access:      local
```

## PKCS#11 Specifications

### Unsupported Key Types and Mechanisms

- Sign Recover and Verify Recover functions are not supported.

### Supported Key Types

The Azure Cloud HSM PKCS#11 library supports the following key types and sizes.

| Key Type                 | Description                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>RSA</b>               | 1024 - 4096-bit modulus in steps of 256 bits. <ul style="list-style-type: none"> <li>Key generation of 1024 modulus size key is supported only in non-FIPS mode of operation.</li> </ul>                                                                                                                                                                                           |
| <b>ECDSA</b>             | Curves supported:<br>secp192r1 (P-192), secp224r1 (P-224), secp256r1 (P-256), secp256k1 (blockchain), secp384r1 (P-384), and secp521r1 (P-521) for key generation. <ul style="list-style-type: none"> <li>Only Curves P-256 and P-384 are supported for sign/verify.</li> <li>P-192 and P-521 for ECDH key derivation are supported only in non-FIPS mode of operation.</li> </ul> |
| <b>AES</b>               | 128, 192, and 256 bits.                                                                                                                                                                                                                                                                                                                                                            |
| <b>Triple DES (3DES)</b> | 192 bits                                                                                                                                                                                                                                                                                                                                                                           |
| <b>DES</b>               | 64 bits <ul style="list-style-type: none"> <li>Supported only in non-FIPS mode of operation.</li> </ul>                                                                                                                                                                                                                                                                            |
| <b>RC4</b>               | 1 - 256 bytes <ul style="list-style-type: none"> <li>Supported only in non-FIPS mode of operation.</li> </ul>                                                                                                                                                                                                                                                                      |
| <b>GENERIC_SECRET</b>    | 1 – 3072 bytes                                                                                                                                                                                                                                                                                                                                                                     |

## Supported Mechanisms

To invoke a cryptographic feature using PKCS#11, call a function with a given mechanism. The following sections summarize the combinations of functions and mechanisms supported by Azure Cloud HSM.

The Azure Cloud HSM PKCS#11 library supports the following algorithms:

- **Encryption and decryption** – AES-CBC, AES-CTR, AES-ECB, AES-GCM, DES3-CBC, DES3-ECB, RSA-OAEP, and RSA-PKCS
- **Sign and verify** – RSA, HMAC, and ECDSA, with and without hashing.
- **Hash/digest** – SHA1, SHA224, SHA256, SHA384, and SHA512
- **Key wrap** – AES Key Wrap, AES-GCM, RSA-AES, and RSA-OAEP

## Supported Key Generation Mechanisms

| Mechanism       | Min Key Length | Max Key Length | Algorithm | Mode   |
|-----------------|----------------|----------------|-----------|--------|
| CKM_RC4_KEY_GEN | 1 byte         | 256 bytes      | RC4       | STREAM |



|                            |           |           |       |      |
|----------------------------|-----------|-----------|-------|------|
| CKM_GENERIC_SECRET_KEY_GEN | 1 byte    | 800 bytes | None  | None |
| CKM_DES3_KEY_GEN           | 192 bits  | 192 bits  | DES3  | None |
| CKM_AES_KEY_GEN            | 128 bits  | 256 bits  | AES   | None |
| CKM_RSA_X9_31_KEY_PAIR_GEN | 2048 bits | 4098 bits | RSA   | None |
| CKM_RSA_PKCS_KEY_PAIR_GEN  | 2048 bits | 4098 bits | RSA   | None |
| CKM_EC_KEY_PAIR_GEN        | 256 bits  | 521 bits  | ECDSA | None |

*\* The mechanism CKM\_EC\_KEY\_PAIR\_GEN must be used to generate Ed25519 and X25519 key pairs*

## Supported Key Encryption/Decryption Mechanisms

| Mechanism                  | Min Key Length | Max Key Length | Algorithm | Mode    |
|----------------------------|----------------|----------------|-----------|---------|
| CKM_DES_CBC                | 64 bits        | 64 bits        | DES       | CBC     |
| CKM_DES_CBC_PAD            | 64 bits        | 64 bits        | DES       | CBC_PAD |
| CKM_DES3_CBC               | 192 bits       | 192 bits       | DES3      | CBC     |
| CKM_DES3_CBC_PAD           | 192 bits       | 192 bits       | DES3      | CBC_PAD |
| CKM_DES3_ECB               | 192 bits       | 192 bits       | DES3      | ECB     |
| CKM_RC4                    | 8 bits         | 2048 bits      | RC4       | None    |
| CKM_AES_CBC                | 128 bits       | 256 bits       | AES       | CBC     |
| CKM_AES_CBC_PAD            | 128 bits       | 256 bits       | AES       | CBC_PAD |
| CKM_AES_ECB                | 128 bits       | 256 bits       | AES       | ECB     |
| CKM_AES_CTR                | 128 bits       | 256 bits       | AES       | CTR     |
| CKM_AES_GCM                | 128 bits       | 256 bits       | AES       | GCM     |
| CKM_CLOUDHSM_AES_GCM       | 128 bits       | 256 bits       | *         | GCM     |
| CKM_RSA_PKCS               | 2048 bits      | 4098 bits      | RSA       | None    |
| CKM_RSA_PKCS_OAEP          | 2048 bits      | 4098 bits      | RSA       | None    |
| CKM_AES_KEY_WRAP           | 128 bits       | 256 bits       | AES       | None    |
| CKM_AES_KEY_WRAP_PAD       | 128 bits       | 256 bits       | AES       | None    |
| CKM_AES_KEY_WRAP_NO_PAD    | 128 bits       | 256 bits       | AES       | None    |
| CKM_AES_KEY_WRAP_PKCS5_PAD | 128 bits       | 256 bits       | AES       | None    |

*\* The mechanism CKM\_DES\_CBC, CKM\_DES\_CBC\_PAD and CKM\_RC4 are supported in non-FIPS mode only. CKM\_CLOUDHSM\_AES\_GCM is a proprietary mechanism and programmatically safer alternative to the standard CKM\_AES\_GCM.*

## Supported Sign/Verify Mechanisms

| Mechanism               | Min Key Length | Max Key Length | Algorithm | Mode   |
|-------------------------|----------------|----------------|-----------|--------|
| CKM_MD5_RSA_PKCS        | 1024 bits      | 4098 bits      | RSA       | None   |
| CKM_SHA1_RSA_PKCS       | 2048 bits      | 4098 bits      | RSA       | SHA1   |
| CKM_SHA224_RSA_PKCS     | 2048 bits      | 4098 bits      | RSA       | SHA224 |
| CKM_SHA256_RSA_PKCS     | 2048 bits      | 4098 bits      | RSA       | SHA256 |
| CKM_SHA384_RSA_PKCS     | 2048 bits      | 4098 bits      | RSA       | SHA384 |
| CKM_SHA512_RSA_PKCS     | 2048 bits      | 4098 bits      | RSA       | SHA512 |
| CKM_RSA_PKCS_PSS        | 2048 bits      | 4098 bits      | RSA       | None   |
| CKM_SHA1_RSA_PKCS_PSS   | 2048 bits      | 4098 bits      | RSA       | SHA1   |
| CKM_SHA224_RSA_PKCS_PSS | 2048 bits      | 4098 bits      | RSA       | SHA224 |
| CKM_SHA256_RSA_PKCS_PSS | 2048 bits      | 4098 bits      | RSA       | SHA256 |
| CKM_SHA384_RSA_PKCS_PSS | 2048 bits      | 4098 bits      | RSA       | SHA384 |
| CKM_SHA512_RSA_PKCS_PSS | 2048 bits      | 4098 bits      | RSA       | SHA512 |
| CKM_ECDSA               | 256 bits       | 384 bits       | ECDSA     | None   |
| CKM_ECDSA_SHA224        | 256 bits       | 384 bits       | ECDSA     | SHA224 |
| CKM_ECDSA_SHA256        | 256 bits       | 384 bits       | ECDSA     | SHA256 |
| CKM_ECDSA_SHA384        | 256 bits       | 384 bits       | ECDSA     | SHA384 |
| CKM_ECDSA_SHA512        | 256 bits       | 384 bits       | ECDSA     | SHA512 |
| CKM_SHA_1_HMAC          | 8 bits         | 800 bits       | SHA1      | HMAC   |
| CKM_SHA224_HMAC         | 8 bits         | 800 bits       | SHA224    | HMAC   |
| CKM_SHA256_HMAC         | 8 bits         | 800 bits       | SHA256    | HMAC   |
| CKM_SHA384_HMAC         | 8 bits         | 800 bits       | SHA384    | HMAC   |
| CKM_SHA512_HMAC         | 8 bits         | 800 bits       | SHA512    | HMAC   |

\* The mechanism CKM\_MD5\_RSA\_PKCS is supported in non-FIPS mode only.

## Supported Key Hash/Digest Mechanisms

| Mechanism  | Min Key Length | Max Key Length | Algorithm | Supported Functions |
|------------|----------------|----------------|-----------|---------------------|
| CKM_SHA_1  | N/A            | N/A            | SHA       | Digest              |
| CKM_SHA224 | N/A            | N/A            | SHA224    | Digest              |
| CKM_SHA256 | N/A            | N/A            | SHA256    | Digest              |

|            |     |     |        |        |
|------------|-----|-----|--------|--------|
| CKM_SHA384 | N/A | N/A | SHA384 | Digest |
| CKM_SHA512 | N/A | N/A | SHA512 | Digest |

## Supported Key Wrap Mechanisms

| Mechanism                  | Min Key Length | Max Key Length | Algorithm | Mode        |
|----------------------------|----------------|----------------|-----------|-------------|
| CKM_AES_CBC                | 128 bits       | 256 bits       | AES       | Keywrap     |
| CKM_AES_CBC_PAD            | 128 bits       | 256 bits       | AES       | Keywrap     |
| CKM_AES_GCM                | 128 bits       | 256 bits       | AES       | Keywrap     |
| CKM_AES_KEY_WRAP           | 128 bits       | 256 bits       | AES       | Keywrap     |
| CKM_AES_KEY_WRAP_PAD       | 128 bits       | 256 bits       | AES       | Keywrap_pad |
| CKM_AES_KEY_WRAP_NO_PAD    | 128 bits       | 256 bits       | AES       | Keywrap     |
| CKM_AES_KEY_WRAP_PKCS5_PAD | 128 bits       | 256 bits       | AES       | Keywrap     |
| CKM_RSA_AES_KEY_WRAP       | 2048 bits      | 4098 bits      | RSA       | Keywrap     |
| CKM_DES3_NIST_WRAP         | 192 bits       | 192 bits       | DES       | None        |
| CKM_RSA_PKCS               | 2048 bits      | 4096 bits      | AES       | Keywrap     |
| CKM_RSA_PKCS_OAEP          | 2048 bits      | 4096 bits      | AES       | Keywrap     |

*\* The mechanism CKM\_AES\_CBC and CKM\_AES\_CBC\_PAD support Unwrap in FIPS mode, and Wrap/Unwrap in non-FIPS mode.*

## Supported Key Derivation Mechanisms

The Azure Cloud HSM PKCS#11 library allows you to use the following mechanisms for Derive functions.

- CKM\_ECDH1\_DERIVE
- CKM\_SP800\_108\_COUNTER\_KDF

## Supported API Operations

The Azure Cloud HSM PKCS#11 library supports the following PKCS#11 API operations.

| Function        | PKCS#11 API  | Description                                            |
|-----------------|--------------|--------------------------------------------------------|
| General Purpose | C_Initialize | Initializes the key                                    |
|                 | C_finalize   | Cleans up miscellaneous Cryptoki-associated resources. |

|                                  |                     |                                                                                                                   |
|----------------------------------|---------------------|-------------------------------------------------------------------------------------------------------------------|
|                                  | C_GetInfo           | Obtains general information about Cryptoki.                                                                       |
|                                  | C_GetFunctionList   | Obtains the entry point.                                                                                          |
| <b>Slot and Token Management</b> | C_GetSlotInfo       | Obtains information about a particular slot.                                                                      |
|                                  | C_GetTokenInfo      | Obtains information about a particular token.                                                                     |
|                                  | C_GetMechanismList  | Obtains a list of mechanisms supported by a token.                                                                |
|                                  | C_GetMechanismInfo  | Obtains information about a particular mechanism.                                                                 |
|                                  | C_OpenSession       | Opens a connection between an application and a particular token or set up an application call azcloudhsm_client. |
| <b>Session Management</b>        | C_CloseSession      | Close a session                                                                                                   |
|                                  | C_CloseAllSessions  | Close all sessions with a token.                                                                                  |
|                                  | C_Login             | Logs in to a token.                                                                                               |
|                                  | C_Logout            | Logs out from a token.                                                                                            |
|                                  | C_GetSessionInfo    | Gets session information.                                                                                         |
|                                  | C_DestroySession    | Destroys a session.                                                                                               |
| <b>Object Management</b>         | C_CreateObject      | Creates an object. (e.g. certificate object)                                                                      |
|                                  | C_CopyObject        | Copies an object. (e.g. certificate object)                                                                       |
|                                  | C_DestroyObject     | Destroys an object. (e.g. certificate object)                                                                     |
|                                  | C_GetAttributeValue | Obtains an attribute value of an object.                                                                          |
|                                  | C_SetAttributeValue | Updates the value of one or more attributes of a object.                                                          |
|                                  | C_FindObjectsInit   | Initializes an object search operation.                                                                           |
|                                  | C_FindObjects       | Continues an object search operation.                                                                             |
|                                  | C_FindObjectsFinal  | Finishes an object search operation.                                                                              |
| <b>Encryption</b>                | C_EncryptInit       | Initializes an encryption operation                                                                               |
|                                  | C_Encrypt           | Encrypts single-part data.                                                                                        |
|                                  | C_EncryptUpdate     | Continues a multiple-part encryption operation                                                                    |
|                                  | C_EncryptFinal      | Finishes a multiple-part encryption operation.                                                                    |
| <b>Decryption</b>                | C_DecryptInit       | Initializes a decryption operation.                                                                               |
|                                  | C_Decrypt           | Decrypts single-part data.                                                                                        |
|                                  | C_DecryptUpdate     | Continues a multiple-part decryption operation.                                                                   |
|                                  | C_DecryptFinal      | Finishes a multiple-part decryption operation.                                                                    |
| <b>Message Digest</b>            | C_DigestInit        | Initializes a message-digesting operation.                                                                        |
|                                  | C_Digest            | Digests single-part data.                                                                                         |
|                                  | C_DigestUpdate      | Continues a multiple-part digesting operation.                                                                    |
|                                  | C_DigestFinal       | Finishes a multiple-part digesting operation.                                                                     |

|                                     |                     |                                                                                           |
|-------------------------------------|---------------------|-------------------------------------------------------------------------------------------|
| <b>Signing and MAC</b>              | C_SignInit          | Initializes a signature operation.                                                        |
|                                     | C_Sign              | Signs single-part data.                                                                   |
|                                     | C_SignUpdate        | Continues a multiple-part signature operation.                                            |
|                                     | C_SignFinal         | Finishes a multiple-part signature operation.                                             |
|                                     | C_SignRecoverInit   | Initializes a signature operation, where the data can be recovered from the signature.    |
|                                     | C_SignRecover       | Signs single-part data, where the data can be recovered from the signature.               |
| <b>Verifying Signatures and MAC</b> | C_VerifyInit        | Initializes a verify operation.                                                           |
|                                     | C_Verify            | Verifies a signature on single-part data.                                                 |
|                                     | C_VerifyUpdate      | Continues a multiple part verify operation                                                |
|                                     | C_VerifyFinal       | Finishes a multiple-part verification operation.                                          |
|                                     | C_VerifyRecoverInit | Initializes a verification operation where the data is recovered from the signature       |
|                                     | C_VerifyRecover     | Verifies a signature on single-part data, where the data is recovered from the signature. |
| <b>Random Number Generators</b>     | C_SeedRandom        | Mixes in additional seed material to the random number generator.                         |
|                                     | C_GenerateRandom    | Generates random data.                                                                    |
| <b>Parallel Functions</b>           | C_GetFunctionStatus | Obtains updated status of a function running in parallel with an application.             |
|                                     | C_CancelFunction    | Cancels a function running in parallel.                                                   |
| <b>Key Management</b>               | C_GenerateKey       | Generates a secret key.                                                                   |
|                                     | C_GenerateKeyPair   | Generates a public-key/private-key pair.                                                  |
|                                     | C_WrapKey           | Wraps (encrypts) a key.                                                                   |
|                                     | C_UnwrapKey         | Unwarps (decrypts) a key.                                                                 |
|                                     | C_DeriveKey         | Derives a key from a base key.                                                            |

## Supported Attributes

A key object can be a public, private, or secret key. Actions permitted on a key object are specified through attributes. Attributes are defined when the key object is created. When you use the Azure Cloud HSM PKCS#11 library, we assign default values as specified by the PKCS#11 standard.

Azure Cloud HSM does not support all attributes listed in the PKCS#11 specification. We are compliant with the specification for all attributes we support. These attributes are listed in the respective tables.

The PKCS#11 library table contains a list of attributes that differ by key types. It indicates whether a given attribute is supported for a particular key type when using a specific cryptographic function with Azure Cloud HSM.

| Attribute          | Firmware Stores | Description                                                                                                                                                                   | Can be changed after object creation. | Default Value |
|--------------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|---------------|
| CKA_SIGN           | Yes             | Boolean type CK_TRUE if key supports signatures                                                                                                                               | Yes                                   | FALSE         |
| CKA_SIGN_RECOVER   | Yes             | CK_TRUE if key supports signatures where the data can be recovered from the signature. Only applicable to private keys. If ND (Not Defined), value is inherited from CKA_SIGN | Yes                                   | FALSE         |
| CKA_VERIFY         | Yes             | Boolean type CK_True if key supports verification.                                                                                                                            | Yes                                   | FALSE         |
| CKA_VERIFY_RECOVER | Yes             | CK_TRUE if key supports verification where data is recovered from the signature. Only applicable to public keys. If ND, value is inherited from CKA_SIGN                      | Yes                                   | FALSE         |
| CKA_ENCRYPT        | Yes             | Boolean type. CK_TRUE if key supports encryption.                                                                                                                             | Yes                                   | FALSE         |
| CKA_DECRYPT        | Yes             | Boolean type. CK_TRUE if key supports decryption.                                                                                                                             | Yes                                   | FALSE         |
| CKA_WRAP           | Yes             | Boolean type. CK_TRUE if key supports wrapping.                                                                                                                               | Yes                                   | FALSE         |
| CKA_UNWRAP         | Yes             | Boolean type. CK_TRUE if key supports unwrapping.                                                                                                                             | Yes                                   | FALSE         |
| CKA_LABEL          | Yes             | Description of the object.                                                                                                                                                    | Yes                                   | NA            |
| CKA_ID             | Yes             | Byte array; key identifier for key.                                                                                                                                           | No                                    | NA            |
| CKA_CHECK_VALUE    | Yes             | Byte array; KeyChecksum                                                                                                                                                       | No                                    | NA            |
| CKA_CLASS          | Yes             | Object class                                                                                                                                                                  | No                                    | NA            |

|                       |     |                                                                                                                                                                                                                                                |                           |                                                           |
|-----------------------|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|-----------------------------------------------------------|
| CKA_LOCAL             | Yes | Boolean type. TRUE is key was generated locally.                                                                                                                                                                                               | No                        | Derived                                                   |
| CKA_EXTRACTABLE       | Yes | Boolean type. TRUE if key is extractable and can be wrapped.                                                                                                                                                                                   | No                        | TRUE                                                      |
| CKA_EC_PARAMS         | Yes |                                                                                                                                                                                                                                                | No                        | NA                                                        |
| CKA_EC_POINT          | Yes | Supported only for create public key.                                                                                                                                                                                                          | No                        | NA                                                        |
| CKA_PRIVATE_EXPONENT  | Yes | Private exponent d. Supported only for create private object.                                                                                                                                                                                  | No                        | NA                                                        |
| CKA_SENSITIVE         | Yes | TRUE if key is sensitive.                                                                                                                                                                                                                      | Yes                       | TRUE (returns the value from firmware)                    |
| CKA_DERIVE            | Yes | CK_TRUE if key supports key derivation. This attribute is used to derive another key from the input key. This attribute must be FALSE for all public keys and cannot be set to TRUE. For secret and EC private keys, this can be TRUE or FALSE | Yes                       | From firmware<br>TRUE, but<br>from PKCS#11<br>FALSE       |
| CKA_TRUSTED           | Yes | Can be set to TRUE or FALSE by CO only. The key can be trusted for the application that it was created.                                                                                                                                        | Yes                       | FALSE                                                     |
| CKA_NEVER_EXTRACTABLE | Yes | Returns the value set by the firmware.                                                                                                                                                                                                         | No                        | TRUE for private and secret keys<br>FALSE for public keys |
| CKA_ALWAYS_SENSITIVE  | Yes | CK_TRUE if key has always had the CKA_SENSITIVE attribute set to CK_TRUE.                                                                                                                                                                      | No                        | TRUE                                                      |
| CKA_WRAP_WITH_TRUSTED | Yes | Can be set from FALSE to TRUE but cannot be changed after set to TRUE. CK_TRUE if the key can only be                                                                                                                                          | Depends on initial value. | FALSE                                                     |

|                         |     |                                                                                                                                                                                                        |     |                                               |
|-------------------------|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|-----------------------------------------------|
|                         |     | wrapped with a wrapping key that has CKA_TRUSTED set to CK_TRUE.                                                                                                                                       |     |                                               |
| CKA_DESTROYABLE         | Yes | Can be set to TRUE or FALSE. CK_TRUE if the object can be destroyed using C_DestroyObject. Note this attribute is not applied to session keys. This is a deviation from the specification.             | Yes | TRUE                                          |
| CKA_WRAP_TEMPLATE       | Yes | his attribute is not applied to session keys. This is a deviation from the specification.                                                                                                              | Yes | Not defined (returns the value from firmware) |
| CKA_UNWRAP_TEMPLATE     | Yes | For wrapping keys. The attribute template to apply to any keys unwrapped using this wrapping key. Any user-supplied template is applied after this template as if the object has already been created. | Yes | Not defined (returns the value from firmware) |
| CKA_SUBJECT             | No  | DER-encoding of the certificate subject name.                                                                                                                                                          | No  | NA                                            |
| CKA_OBJECT_ID           | No  | DER-encoding of the object identifier indicating the data object type.                                                                                                                                 | No  | NA                                            |
| CKA_APPLICATION         | No  | Description of the application that manages the object.                                                                                                                                                | No  | NA                                            |
| CKA_START_DATE          | No  | Start date for the key.                                                                                                                                                                                | No  | NA                                            |
| CKA_END_DATE            | No  | End date for the key.                                                                                                                                                                                  | No  | NA                                            |
| CKA_KEY_GEN_MECHANISM   | No  | Identifier for the mechanism used to generate the key material.                                                                                                                                        | No  | NA                                            |
| CKA_ALLOWED_MECHANISMS  | No  | A list of mechanisms allowed to be used with this key                                                                                                                                                  | No  | NA                                            |
| CKA_ALWAYS_AUTHENTICATE | No  | If CK_TRUE, the user must supply the PIN for each use (sign or decrypt) with the key.                                                                                                                  | No  | NA                                            |
| CKA_MODIFIABLE          | No  | CK_TRUE if object can be modified; default is CK_TRUE                                                                                                                                                  | No  | NA                                            |



|                     |     |                                                                                            |    |       |
|---------------------|-----|--------------------------------------------------------------------------------------------|----|-------|
| CKA_PUBLIC_KEY_INFO | Yes | DER-encoding of the SubjectPublicKeyInfo for the public key contained in this certificate. | No | NA    |
| CKA_COPYABLE        | No  | CK_TRUE if object can be copied using C_CopyObject. Defaults to CK_TRUE                    | No | NA    |
| CKA_PRIME_1         | No  | Prime p.                                                                                   | No | NA    |
| CKA_PRIME_2         | No  | Prime q.                                                                                   | No | NA    |
| CKA_EXPONENT_1      | No  | Private exponent d modulo p-1.                                                             | No | NA    |
| CKA_EXPONENT_2      | No  | Private exponent d modulo q-1.                                                             | No | NA    |
| CKA_PRIME           | No  |                                                                                            | No | NA    |
| CKA_SUBPRIME        | No  |                                                                                            | No | NA    |
| CKA_BASE            | No  |                                                                                            | No | NA    |
| CKA_PRIME_BITS      | No  |                                                                                            | No | NA    |
| CKA_SUBPRIME_BITS   | No  |                                                                                            | No | NA    |
| CKA_SUB_PRIME_BIT   | No  |                                                                                            | No | NA    |
| CKA_TOKEN           | No  | CK_TRUE if object is a token object; CK_FALSE if object is a session object                | No | FALSE |
| CKA_VALUE_BITS      | No  |                                                                                            | No | NA    |
| CKA_VALUE_LEN       | No  |                                                                                            | No | NA    |
| CKA_DERIVE_TEMPLATE | No  |                                                                                            | No | NA    |
| CKA_MECHANISM_TYPE  | No  |                                                                                            | No | NA    |

## Support for Certificate Storage

Azure Cloud HSM provides robust support for certificate storage using the PKCS#11 API. The Azure Cloud HSM PKCS#11 library supports the following PKCS#11 API operations.

- C\_CreateObject: Creates a new certificate object.
- C\_DestroyObject: Deletes an existing certificate object.
- C\_CopyObject: Copies an existing certificate object.
- C\_GetAttributeValue: Gets the value of one or more attributes of a certificate object.
- C\_SetAttributeValue: Updates the value of one or more attributes of a certificate object.
- C\_FindObjectsInit: Starts a search for certificate objects.
- C\_FindObjects: Continues a search for certificate objects.

- C\_FindObjectsFinal: Ends a search for certificate objects.