THALES

Thales Luna Network HSM 7

UTILITIES REFERENCE

Document Information

Last Updated	2023-05-25 23:51:04 GMT-04:00

Disclaimer
All information herein is either public information or is the property of and owned solely by Thales Group and/or its subsidiaries who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Thales Group's information.

This document can be used for informational, non-commercial, internal, and personal use only provided that:

- The copyright notice, the confidentiality and proprietary legend and this full warning notice appear in all copies.
- This document shall not be posted on any publicly accessible network computer or broadcast in any media, and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided "AS IS" without any warranty of any kind. Unless otherwise expressly agreed in writing, Thales Group makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Thales Group reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Thales Group hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Thales Group be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Thales Group does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Thales Group be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Thales products. Thales Group disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed

that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service, or loss of privacy.

Regulatory Compliance

This product complies with the following regulatory regulations. To ensure compliancy, ensure that you install the products as specified in the installation instructions and use only Thales-supplied or approved accessories.

USA, FCC

This equipment has been tested and found to comply with the limits for a “Class B” digital device, pursuant to part 15 of the FCC rules.

Canada

This class B digital apparatus meets all requirements of the Canadian interference-causing equipment regulations.

Europe

This product is in conformity with the protection requirements of EC Council Directive 2014/30/EU. This product satisfies the CLASS B limits of EN55032.

CONTENTS

Preface: About the Utilities Reference 6
Customer Release Notes 6
Audience 6
Document Conventions 7
Support Contacts 9

Chapter 1: cmu 10
Authentication 10
Common CMU Options 10
cmu certify 12
cmu delete 16
cmu export 17
cmu generatekeypair 18
cmu getattribute 23
cmu getpkc 25
cmu import 26
cmu importkey 27
cmu list 29
cmu requestcertificate 32
cmu selfsigncertificate 35
cmu setattribute 40
cmu verifyhsm 43
cmu verifypkc 43

Chapter 2: ckdemo 45
AUDIT/LOG Menu Functions 47
CA Menu Functions 48
CLUSTER EXECUTION Menu Functions 50
HIGH AVAILABILITY RECOVERY Menu Functions 50
KEY Menu Functions 51
KEY AUTHORIZATION Menu Functions 52
OBJECT MANAGEMENT Menu Functions 52
OFFBOARD KEY STORAGE Menu Functions 54
OTHERS Menu Functions 54
(90) Options 55
(98) Options 56
PED INFO Menu Functions 58
POLICY Menu Functions 58
SCRIPT EXECUTION Menu Functions 59
SECURITY Menu Functions 59
SRK Menu Functions 60

TOKEN Menu Functions 60

Chapter 3: multitoken 64
Accessing multitoken 64
Syntax 64
Operating Modes 69
Notes 78
Named and User-defined Curves 79
SKS and Per Key Auth 80
Example with LCO role 81

Chapter 4: rbs 83

Chapter 5: salogin 85

Chapter 6: pscp 88

Chapter 7: vtl 90
vtl addCA 92
vtl addServer 93
vtl addServerNoCert 94
vtl cklogsupport 95
vtl createCert 96
vtl createCSR 98
vtl deleteCA 100
vtl deleteServer 101
vtl deleteServerNoCert 102
vtl examineCert 103
vtl fingerprint 105
vtl listCAs 106
vtl listServers 107
vtl listSlots 108
vtl logging 109
vtl replaceServer 110
vtl supportInfo 111
vtl verify 112

PREFACE: About the Utilities Reference

This document describes how to use the various utilities included with the Luna HSM Client. It contains the following chapters:

> "cmu" on page 10
> "ckdemo" on page 45
> "multitoken" on page 64
> "rbs" on page 83
> "salogin" on page 85
> "pscp" on page 88
> "vtl" on page 90

The preface includes the following information about this document:

> "Customer Release Notes" below
> "Audience" below
> "Document Conventions" on the next page
> "Support Contacts" on page 9

For information regarding the document status and revision history, see "Document Information" on page 2.

Customer Release Notes

The Customer Release Notes (CRN) provide important information about specific releases. Read the CRN to fully understand the capabilities, limitations, and known issues for each release. You can view the latest version of the CRN at www.thalesdocs.com.

Audience

This document is intended for personnel responsible for maintaining your organization's security infrastructure. This includes Luna HSM users and security officers, key manager administrators, and network administrators.

All products manufactured and distributed by Thales are designed to be installed, operated, and maintained by personnel who have the knowledge, training, and qualifications required to safely perform the tasks assigned to them. The information, processes, and procedures contained in this document are intended for use by trained and qualified personnel only.

It is assumed that the users of this document are proficient with security concepts.

Document Conventions

This document uses standard conventions for describing the user interface and for alerting you to important information.

Notes

Notes are used to alert you to important or helpful information. They use the following format:

> **NOTE** Take note. Contains important or helpful information.

Cautions

Cautions are used to alert you to important information that may help prevent unexpected results or data loss. They use the following format:

> **CAUTION!** Exercise caution. Contains important information that may help prevent unexpected results or data loss.

Warnings

Warnings are used to alert you to the potential for catastrophic data loss or personal injury. They use the following format:

> ****WARNING** Be extremely careful and obey all safety and security measures. In this situation you might do something that could result in catastrophic data loss or personal injury.**

Command syntax and typeface conventions

Format	Convention
bold	The bold attribute is used to indicate the following: > Command-line commands and options (Type **dir /p**.) > Button names (Click **Save As**.) > Check box and radio button names (Select the **Print Duplex** check box.) > Dialog box titles (On the **Protect Document** dialog box, click **Yes**.) > Field names (**User Name**: Enter the name of the user.) > Menu names (On the **File** menu, click **Save**.) (Click **Menu** > **Go To** > **Folders**.) > User input (In the **Date** box, type **April 1**.)
italics	In type, the italic attribute is used for emphasis or to indicate a related document. (See the *Installation Guide* for more information.)

Format	Convention
<variable>	In command descriptions, angle brackets represent variables. You must substitute a value for command line arguments that are enclosed in angle brackets.
[**optional**] [<optional>]	Represent optional **keywords** or <variables> in a command line description. Optionally enter the keyword or <variable> that is enclosed in square brackets, if it is necessary or desirable to complete the task.
{**a**|**b**|**c**} {<a>|<b>|<c>}	Represent required alternate **keywords** or <variables> in a command line description. You must choose one command line argument enclosed within the braces. Choices are separated by vertical (OR) bars.
[**a**|**b**|**c**] [<a>|<b>|<c>]	Represent optional alternate keywords or variables in a command line description. Choose one command line argument enclosed within the braces, if desired. Choices are separated by vertical (OR) bars.

Support Contacts

If you encounter a problem while installing, registering, or operating this product, please refer to the documentation before contacting support. If you cannot resolve the issue, contact your supplier or Thales Customer Support. Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access is governed by the support plan negotiated between Thales and your organization. Please consult this plan for details regarding your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is where you can find solutions for most common problems and create and manage support cases. It offers a comprehensive, fully searchable database of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more.

> **NOTE** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

Telephone

The support portal also lists telephone numbers for voice contact (Contact Us).

CHAPTER 1: cmu

> **NOTE** This is a general-purpose tool intended for use across Luna HSM versions. It might reference mechanisms and features that are not available on all Luna products.

This section provides a detailed description of each function available in the Certificate Management Utility.

The command function is the first parameter on the command line that invokes the CMU application. It does not require a leading dash character. All options follow the command function and do employ leading dashes. Only a single command function can be specified with each invocation of the CMU application.

```
cmu <function> <-parameter_name[=parameter_value]>
```

Most functions take parameters, some of which may be mandatory, and some optional. Parameters may, in turn, take values. If a parameter takes a value, then the general syntax is to write the command **cmu**, followed by a space, followed by a function name, followed by a space, followed by a leading dash "-" and parameter name and an equal sign "=" and a value, with no spaces from the dash to the end of the parameter value. Multiple parameters are separated by spaces.

Authentication

Where an operation requires authentication, you must provide the appropriate password (for a password-authenticated HSM) or the appropriate PED key (via Luna PED, for a multifactor quorum-authenticated HSM).

Common CMU Options

Some options are commonly available to all **cmu** commands. They are described below and not on the individual command pages, for conciseness.

Argument(s)	Description
-cu	Specifies that you wish to perform the command as the partition's Crypto User. If the CU is not authorized to perform the operation, the command fails. If a role is not specified, the Crypto Officer role is used by default. Requires minimum Luna HSM Client 10.4.0.
-lco	Specifies that you wish to perform the command as the partition's Limited Crypto Officer. If the LCO is not authorized to perform the operation, the command fails. If a role is not specified, the Crypto Officer role is used by default. Requires minimum Luna HSM Firmware 7.7.0 and minimum Luna HSM Client 10.3.0.
-password=<password>	The password for the role accessing the current slot, with the current command. If this is not specified, it is prompted.

Argument(s)	Description
-ped=<PED_ID>	Specifies the PED ID for the registered Remote PED that will handle authentication for the current slot, with the current command. You must specify this parameter to use Remote PED authentication.
-slot=<slot#>	The slot to be acted upon, by the current command. If this is not specified, it is prompted.
-so	Specifies that you wish to perform the command as Partition Security Officer for that slot. If a role is not specified, the Crypto Officer role is used by default.

This chapter provides a detailed description of each of the functions available in the Luna Certificate Management Utility. It contains the following topics:

> "cmu certify" on the next page
> "cmu delete" on page 16
> "cmu export" on page 17
> "cmu generatekeypair" on page 18
> "cmu getattribute" on page 23
> "cmu getpkc" on page 25
> "cmu import" on page 26
> "cmu importkey" on page 27
> "cmu list" on page 29
> "cmu requestcertificate" on page 32
> "cmu selfsigncertificate" on page 35
> "cmu setattribute" on page 40
> "cmu verifyhsm" on page 43
> "cmu verifypkc" on page 43

cmu certify

This function creates an X.509 V3 certificate from a PKCS #10 certificate request. The parent certificate and corresponding private key must already exist on the token or HSM. The private key is located on the token using the public key information inside the parent certificate.

> **NOTE** This command requires DER encoding for certificate requests generated outside of cmu. Both DER and PEM encoding are acceptable for certificate requests generated through "cmu requestcertificate" on page 32.

Syntax

cmu certify {**-handle**=<handle#> | **-ouid**=<OUID#>} **-inputfile**=<filename> **-startDate**=<YYYYMMDD> **-endDate**=<YYYYMMDD> [**-label**=<label>] [**-id**=<CKA_ID>] [**-certificatepolicy**=<policy>] [**-private**=<**T**/**F**>] [**-keyids**=<value>] [**-keyidalg**=<algorithm>] [**-binary**] [**-keyusage**=<extension(s)>] [**-md5WithRsa**] [**-sha1WithRsa**] [**-sha224withrsa**] [**-sha256withrsa**] [**-sha384withrsa**] [**-sha512withrsa**] [**-sha1withdsa**] [**-sha1withecdsa**] [**-sha224withecdsa**] [**-sha256withecdsa**] [**-sha384withecdsa**] [**-sha512withecdsa**] [**-basicconstraints**=<constraints>] [**-certdelete**] [**-outputfile**=<filename>] [**-parentlabel**=<label>]

Argument(s)	Description
-basicconstraints =<constraints>	Defines constraints applied to the certificate. Can include one or more in a comma-delimited list. **Valid Values:** critical,optional,ca:true,ca:false,pathlen:[value < 127]
-binary	Defines the created certificate format to be raw binary (DER encoding) instead of the default PEM (base64) encoding.
-certdelete	Use this option when you also specify **-outputfile**, so that the cert goes directly to the file system and is not stored on the HSM; otherwise, a certificate is created and stored internally.
-certificatepolicy=<policy>	Defines the certificate policy to be used.
-endDate =<YYYYMMDD>	Defines the validity end of the certificate, in the format YYYYMMDD.
-extendedkeyusage=<usage>	Defines the permitted additional usage of the key. Can include one or more in a comma-delimited list. **Valid Values:** critical,optional,clientauth,serverauth,codesigning,emailprotection,timestamping,ocspsigningD

Argument(s)	Description
-handle =<handle#>	Defines the handle of the parent certificate. If this parameter is omitted and there is only one certificate on the partition, that certificate is automatically selected. If this parameter is omitted and there are multiple certificates on the partition, the user is prompted to select the certificate. This method of selection applies to Luna HSMs only. On a Luna Cloud HSM service slot, use **-ouid** instead.
-id=<CKA_ID>	Defines the CKA_ID attribute for the certificate object that gets created on the HSM. If omitted, the CKA_ID attribute of the private key is used instead.
-inputfile =<filename>	Defines the name of the file that contains the PKCS #10 certificate request.
-keyidalg =<algorithm>	Specifies the hashing algorithm used to create the subject key identifier (SKI) and authority key identifier (AKI) of the newly created certificate. This option is used with **-keyids**. **Valid values:** > **sha1** > **sha224** > **sha256** > **sha384** > **sha512** **NOTE** This parameter is only available if you are using a Luna Network HSM with Luna HSM Client 10.3.0 and newer.
-keyids =<value>	Indicates whether the newly created certificate will have an SKI and AKI. **NOTE** The usage of this parameter varies by Luna HSM Client. Note the following: > If you are using Luna HSM Client 10.3.0 and newer, the SKI is created using a hashing algorithm while the AKI is either taken from the parent certificate (if the parent certificate already has an AKI) or created using a hashing algorithm. You can specify the algorithm with **-keyidalg**. If no algorithm is specified with **-keyidalg**, SHA-1 is used. > If you are using Luna HSM Client 10.2.0 and older, the SKI is calculated using SHA-1 while the AKI is either taken from the parent certificate (if the parent certificate already has an AKI) or created using SHA-1. **Valid values:** 1,0 (True or False)

Argument(s)	Description
-keyusage =<extension (s)>	Defines the key usage extension for the certificate. This parameter may appear more than once in the parameter set, to define multiple usages, or it can be used once with a comma-separated list of usage types. **Valid values:** digitalsignature,nonrepudiation,keyencipherment,dataencipherment,keyagreement,keycertsign,crlsign,encipheronly,decipheronly
-label=<label>	Defines the label attribute for the certificate object that gets created on the HSM. If omitted, the common name of the subject DN is used instead.
-md5WithRsa	Defines the signature algorithm for the certificate to be pkcs-1-MD5withRSAEncryption. The default is to use sha1WithRsa.
-ouid=<OUID#>	Defines the Object Unified Identifier (OUID) of the parent certificate. If this parameter is omitted and there is only one certificate on the partition, that certificate is automatically selected. If this parameter is omitted and there are multiple certificates on the partition, the user is prompted to select the certificate. This method of selection requires Luna HSM Client 10.2.0 or newer, and applies to Luna Cloud HSM only. On a Luna HSM slot, use **-handle** instead.
-outputfile =<filename>	Defines the filename for the certificate to be created.
-parentlabel =<label>	Specifies the label attribute for the certificate or key object that is to be used as the parent for the new certificate.
-private=<**T**/**F**>	Defines whether a certificate is created in the private space (default is F). Set **-private=T** to require authentication before applications can use the certificate.
-serialNumber =<hex_SN>	Defines the serial number of the certificate, in big-endian hexadecimal form.
-sha1withdsa	Defines the signature algorithm for the certificate to be pkcs-1-SHA1withDSAEncryption. The default is to use sha1WithRsa.
-sha1withecdsa	Defines the signature algorithm for the certificate to be pkcs-1-SHA1withECDSAEncryption. The default is to use sha1WithRsa.
-sha1WithRsa	Defines the signature algorithm for the certificate to be pkcs-1-SHA1withRSAEncryption. The default is to use sha1WithRsa.

Argument(s)	Description
-sha224withecdsa	Defines the signature algorithm for the certificate to be pkcs-1-SHA224withECDSAEncryption. The default is to use sha1WithRsa.
-sha224withrsa	Defines the signature algorithm for the certificate to be pkcs-1-SHA224withRSAEncryption. The default is to use sha1WithRsa.
-sha256withecdsa	Defines the signature algorithm for the certificate to be pkcs-1-SHA256withECDSAEncryption. The default is to use sha1WithRsa.
-sha256withrsa	Defines the signature algorithm for the certificate to be pkcs-1-SHA256withRSAEncryption. The default is to use sha1WithRsa.
-sha384withecdsa	Defines the signature algorithm for the certificate to be pkcs-1-SHA384withECDSAEncryption. The default is to use sha1WithRsa.
-sha384withrsa	Defines the signature algorithm for the certificate to be pkcs-1-SHA384withRSAEncryption. The default is to use sha1WithRsa.
-sha512withecdsa	Defines the signature algorithm for the certificate to be pkcs-1-SHA512withECDSAEncryption. The default is to use sha1WithRsa.
-sha512withrsa	Defines the signature algorithm for the certificate to be pkcs-1-SHA512withRSAEncryption. The default is to use sha1WithRsa.
-startDate =<YYYYMMDD>	Defines the validity start of the certificate, in the format YYYYMMDD.

See also "Common CMU Options" on page 10.

Example

The following command generate a certificate request with cmu:

```
cmu requestCert –privatehandle=7 -publichandle=6 -C=CA -L=Ottawa -O=Thales -CN=TestCertificate
-outputFile=testCert.req
```

Alternatively, the following command generates a DER-encoded certificate request with OpenSSL:

```
openssl req -new -key privatekey.pem -out testCert.req -subj
'/C=CA/ST=Ontario/L=Ottawa/O=Thales/CN=TestCertificate' -outform DER
```

The following command creates and signs a new certificate from the testCert.req certificate request, using certificate 8 as the parent:

```
cmu certify -input=testCert.req -h=8
```

cmu delete

This function deletes a key, certificate, or generic data object on the token. A confirmation message is presented to the user, describing the class and label of the object about to be deleted.

Syntax

cmu delete { **-handle**=<handle#> | **-ouid**=<OUID#> | **-certlabel**=<label> | **-privatelabel**=<label> | **-publiclabel**=<label> } [**-force**]

Argument(s)	Description
-certlabel=<label>	The label identifying the certificate to delete. Can be used instead of the object handle.
-force	Proceed without prompting for confirmation.
-handle=<handle#>	The handle of the object to be deleted. This method of selection applies to Luna HSMs only. On a Luna Cloud HSM service slot, use **-ouid**.
-ouid=<OUID#>	The Object Unified Identifier (OUID) of the object to be deleted. This method of selection requires Luna HSM Client 10.2.0 or newer, and applies to Luna Cloud HSM services only. On a Luna HSM slot, use **-handle**.
-privatelabel=<label>	The label identifying the private key to delete. Can be used instead of the object handle or OUID.
-publiclabel=<label>	The label identifying the public key to delete. Can be used instead of the object handle or OUID.

See also "Common CMU Options" on page 10.

Example

The following command deletes the key or certificate referenced by object handle 14 without a request for confirmation of the delete operation:

```
cmu delete -handle=14 -force
```

The following command queries the user for a handle of an object to delete. The user is asked to confirm the deletion operation:

```
cmu delete
```

cmu export

This function exports an X.509 certificate or public key from the token or HSM to a file. The supported formats are Raw (binary) and PEM (base 64 encoding).

Syntax

cmu export { **-handle**=<handle#> | **-ouid**=<OUID#> | **-label**=<label> } **-outputfile**=<filename> [**-binary**] [**-key**] [**-certdelete**]

Argument(s)	Description
-binary	Defines the export format as raw binary (DER encoding) instead of the default PEM (base64) encoding.
-certdelete	Specifies that the certificate is to be deleted from the HSM after it is exported (equivalent to running the **cmu delete** command separately).
-handle=<handle#>	The handle of the X.509 certificate to be exported from the HSM to a file. This method of selection applies to Luna HSMs only. On a Luna Cloud HSM service slot, use **-ouid**.
-key	Specifies that the object being exported is a public key.
-label=<label>	The label of the object to export.
-ouid=<OUID#>	The Object Unified Identifier (OUID) of the X.509 certificate to be exported from the HSM to a file. This method of selection requires Luna HSM Client 10.2.0 or newer, and applies to Luna Cloud HSM services only. On a Luna HSM slot, use **-handle**.
-outputfile=<filename>	Defines the name of the file that receives the exported certificate.

See also "Common CMU Options" on page 10.

Example

The following command outputs the certificate with handle 7 to file test.cer in PEM format:

```
cmu export -handle=7 -outputfile=test.cer
```

cmu generatekeypair

This function generates an asymmetric key pair on the token or HSM. An optional input filename can be used to specify a file from which mandatory and optional attributes are to be read.

For DSA key generation, the domain parameters (Prime, Subprime, and Base) are required, and must be provided either as part of the command, or as responses to interactive prompting. If one is provided at the command line, then all three must be provided in that manner. If none are provided at the command line, then all three are prompted for interactive entry.

You may not provide only one or two of the parameters at the command line. Providing just one or two domain parameters is considered an error, and the command halts with an error message.

Syntax

cmu generatekeypair [**-keyType**=<keytype>] [**-modulusBits**=<length>] [**-publicExponent**=<value>] [**-label**=<label>] [**-inputFile**=<filename>] [**-labelPublic**=<label>] [**-labelPrivate**=<label>] [**-mech**]=<pkcs | prime | aux>[**-modifiable**=<0/1>] [**-encrypt**=<0/1>] [**-decrypt**=<0/1>] [**-sign**=<0/1>] [**-verify**=<0/1>] [**-wrap**=<0/1>] [**-unwrap**=<0/1>] [**-extractable**=<0/1>] [**-id**=<hex_ID>] [**-startDate**=<YYYYMMDD>] [**-endDate**=<YYYYMMDD>] [**-subject**=<hex_value>] [**-curvetype**=<value>] [**-prime**=<length>] [**-subprime**=<length>] [**-base**=<length>]

Argument(s)	Description
-base=<length>	Defines a base length for DSA key generation.
-curvetype=<value>	This optional parameter defines the name of a curve type for ECDSA keys. Enter values 1-5 (1-NISTP 192 / 2-NISTP 224 / 3-NISTP 256 / 4-NISTP 384 / 5-NISTP 521).
-decrypt=<0/1>	Defines the decrypt setting for the private key in the newly generated key pair. It must be set to True or False (or 1 or 0), with False being the default. If this parameter is set to True, then the encrypt setting for the public key should also be set to True. **NOTE** An HSM is often configured such that no key can have multiple functions - see policy #10 on Partition Capabilities and Policies to choose that configuration option. Thus if decrypt is set True, then unwrap and sign would need to be False.
-derive=<0/1>	Set to True or False (or 1 or 0). Note that an HSM is typically configured such that functional key attributes cannot be changed, so attempting to change this attribute will be rejected by the HSM.

Argument(s)	Description
-encrypt=<0/1>	Defines the encrypt setting for the public key in the newly generated key pair. It must be set to True or False (or 1 or 0), with False being the default. If this parameter is set to True, then the decrypt setting for the private key should also be set to True. **NOTE** An HSM is often configured such that no key can have multiple functions - see policy #10 on Partition Capabilities and Policies to choose that configuration option. Thus if encrypt is set True, then wrap and verify would need to be False.
-endDate=<YYYYMMDD>	Defines the endDate field for the newly generated keys. The format for the value is YYYYMMDD.
-extractable=<0/1>	Defines the extractable setting for the private key in the newly generated key pair. It must be set to True or False (or 1 or 0), with False being the default.
-id=<hex_ID>	Defines the ID field for the newly generated keys. It must be set to a big-endian hexadecimal integer value.
-inputFile=<filename>	Defines the name of a file from which to obtain additional parameter settings, one per line, of the form <name>=<value>.
-keygenmech=<**1**/**2**/**3**>	Defines the RSA key generation mechanism to be used. Applies to Luna HSM Client 7.3.0 and earlier. For newer versions, use **-mech**. **Valid Values:** > **1** -- PKCS > **2** -- FIPS 186-3 Only Primes > **3** -- FIPS 186-3 Auxiliary Primes
-keyType=<keytype>	Defines the type of asymmetric keys to generate. This parameter is not required if the key type can be established by the presence of other parameters. (e.g. If **-modulusBits** and/or **-publicExponent** parameters are specified, then **-keyType**=RSA is redundant). Currently, only RSA key pairs are supported.
-label=<label>	Defines a label to be applied to both of the newly generated keys. If a multiple word label is required, the label value must be enclosed within quotation marks.
-labelPrivate=<label>	Defines a label to apply to the private key from the newly generated key pair.
-labelPublic=<label>	Defines a label to apply to the public key from the newly generated key pair.

Argument(s)	Description
-mech=<mechanism>	Defines the RSA key generation mechanism to be used (formerly "**-keygenmech**"). Applies to Luna HSM Client 7.4.0 and newer. For previous versions, use **-keygenmech**. **Valid Values:** > **pkcs** -- PKCS > **prime** -- FIPS 186-3 Only Primes > **aux** -- FIPS 186-3 Auxiliary Primes
-modifiable=<0/1>	Defines the modifiable setting for each of the keys in the key pair. It must be set to True or False (or 1 or 0).
-modulusBits=<length>	Defines the length in bits of the modulus value for the generation of RSA key pairs. It must be set to a value between 1024 and 4096 that is a multiple of 64 bits. If the HSM policy 12 "Allow non-FIPS algorithms" is set to "No", then RSA key size is limited to 2048 bits or 3072 bits.
-prime=<length>	Defines a prime length for DSA key generation.
-publicExponent=<value>	Defines the public exponent value to use in the generation of RSA key pairs. **Valid values:** 3,17,65537. Only 65537 is allowed in FIPS mode.
-sign=<0/1>	Defines the sign setting for the private key in the newly generated key pair. It must be set to True or False (or 1 or 0), with False being the default. If this parameter is set to True, then the verify setting for the public key should also be set to True. **NOTE** An HSM is often configured such that no key can have multiple functions - see policy #10 on Partition Capabilities and Policies to choose that configuration option. Thus if sign is set True, then unwrap and decrypt would need to be False.
-startDate=<YYYYMMDD>	Defines the startDate field for the newly generated keys. The format for the value is YYYYMMDD.
-subject=<hex_value>	Defines the subject field for the newly generated keys. It must be set to a big-endian hexadecimal integer value. The subject field is typically set to the DER encoding of the subject distinguished name for the key.
-subprime=<length>	Defines a subprime bits length for DSA key generation.

Argument(s)	Description
-unwrap=<0/1>	Defines the unwrap setting for the private key in the newly generated key pair. It must be set to True or False (or 1 or 0), with False being the default. If this parameter is set to True, then the wrap setting for the public key should also be set to True. **NOTE** An HSM is often configured such that no key can have multiple functions - see policy #10 on Partition Capabilities and Policies to choose that configuration option. Thus if unwrap is set True, then decrypt and sign would need to be False.
-verify=<0/1>	Defines the verify setting for the public key in the newly generated key pair. It must be set to True or False (or 1 or 0), with False being the default. If this parameter is set to True, then the sign setting for the private key should also be set to True. **NOTE** An HSM is often configured such that no key can have multiple functions - see policy #10 on Partition Capabilities and Policies to choose that configuration option. Thus if verify is set True, then encrypt and wrap would need to be False.
-wrap=<0/1>	Defines the wrap setting for the public key in the newly generated key pair. It must be set to True or False (or 1 or 0), with False being the default. If this parameter is set to True, then the unwrap setting for the private key should also be set to True. **NOTE** An HSM is often configured such that no key can have multiple functions - see policy #10 on Partition Capabilities and Policies to choose that configuration option. Thus if wrap is set True, then encrypt and verify would need to be False.

See also "Common CMU Options" on page 10.

Example

RSA

```
C:\Program Files\SafeNet\LunaClient>cmu gen -modulusBits=2048 -publicExp=65537 -sign=T -
verify=T
Select token
[1] Token Label: myPartition1
[2] Token Label: myPartition1
Enter choice: 2
Please enter password for token in slot 2 : ******************
C:\Program Files\SafeNet\LunaClient>cmu list
Select token
[1] Token Label: myPartition1
```

```
[2] Token Label: myPartition1
Enter choice: 2
Please enter password for token in slot 2 : ******************
handle=14       label=NewPublicVerifyingKey
handle=15       label=NewPrivateSigningKey
C:\Program Files\SafeNet\LunaClient>
```

DSA - Domain Parameters at Command Line

```
cmu generatekeypair -keytype DSA -slot 6 -prime
0xfcec6182eb206b43c03e36c0eadabff56a0c2e79def44bc8f2e53699096d1ff270f159785d756921dbff9773ae084
83b662fc07df7512ff68b2e5565fd7982e20c244832aba121cc0799cc09f2d5414d5f3966211365f51b83e9ffcccb3d
88cdf238f7c2739131ca7aadff662fec1fb0e1d311a404260376fd011fe00d0204c3 -subprime
0xd3807353b51c5f71b22ac3d0c7e394148fcedc61 -base
0x42e3778e6ec31b0db07a6b370d7fb6fb4a0bca6deaac371f6adbcbeba38ddf76a47c3c3d79276a0e579ce4e347180
fd9b4ad461d6cf0eac51fb08cf452f624570051e518a75a5bb9c3578a14fd4f27f795b22acea62b1fdf1032c1266da0
81c7fb99c4266626587093fd381617238ee1578fc325548dc1c08e5f9322c3b1205e
```

DSA - Domain Parameters Entered Interactively

```
cmu generatekeypair -keytype DSA -slot 6
The prime, subprime and base values must be entered as a HEX byte array.
For example, to enter a 1024-bit prime value, enter a 128-byte HEX byte array using this
format: 0xa0383ee692f8...
 The prime value can be a 1024-bit, 2048-bit or 3072-bit value.
Enter a prime value:
0xfcec6182eb206b43c03e36c0eadabff56a0c2e79def44bc8f2e53699096d1ff270f159785d7
56921dbff9773ae08483b662fc07df7512ff68b2e5565fd7982e20c244832aba121cc0799cc09f2d5414d5f39662113
65f 51b83e9ffcccb3d88cdf238f7c2739131ca7aadff662fec1fb0e1d311a404260376fd011fe00d0204c3
Enter a 160 bit subprime value: 0xd3807353b51c5f71b22ac3d0c7e394148fcedc61
Enter a 1024-bit base value:
0x42e3778e6ec31b0db07a6b370d7fb6fb4a0bca6deaac371f6adbcbeba38ddf76a47
c3c3d79276a0e579ce4e347180fd9b4ad461d6cf0eac51fb08cf452f624570051e518a75a5bb9c3578a14fd4f27f795
b22 acea62b1fdf1032c1266da081c7fb99c4266626587093fd381617238ee1578fc325548dc1c08e5f9322c3b1205e
```


cmu getattribute

This function outputs any viewable attributes for an object. An optional output filename can be used to direct the output to a file.

Syntax

cmu getAttribute {**-handle**=<handle#> | **-ouid**=<OUID#>} [**-attributes**=<attribute(s)>] [**-outputFile**=<filename>]

Argument(s)	Description
-attributes=<attribute(s)>	Lists the attributes to be displayed for the object as a comma-separated list. Multiple instances of this option can also be used to define multiple attributes. If this parameter is omitted, all viewable attributes are displayed. alwayssensitive application certificatetype class decrypt derive encrypt enddate extractable id issuer keytype label local modifiable modulus modulusbits neverextractable private publicexponent sensitive serialnumber sign startdate subjec token unwrap value verify wrap **TIP** If the object is not secret, its value can be displayed. If the object is secret, then the output of "value" is simply meaningless place-holder data.
-handle=<handle#>	The object handle. If this parameter is omitted and there is only one object on the partition, that object is automatically selected. If this parameter is omitted and there are multiple objects on the partition, you are prompted to select the object. This method of selection applies to Luna HSMs only. On a Luna Cloud HSM service slot, use **-ouid**.
-ouid=<OUID#>	The Object Unified Identifier (OUID). If this parameter is omitted and there is only one object on the partition, that object is automatically selected. If this parameter is omitted and there are multiple objects on the partition, the user is prompted to select the object. This method of selection requires Luna HSM Client 10.2.0 or newer, and applies to Luna Cloud HSM services only. On a Luna HSM slot, use **-handle**.
-outputFile=<filename>	Defines the filename to which the attribute set is written. If this parameter is omitted, the attribute set is written to the display.

See also "Common CMU Options" on page 10.

Example

The following command outputs all of the viewable attributes for the object with handle 46:

```
cmu getAttribute -handle=46
```

The following command outputs the label, public exponent and modulus of key 9 to file **keydata.txt**:

```
cmu getAttribute -handle=9 -attribute=label,publicExponent,modulus -outputFile=keydata.txt
```

cmu getpkc

Retrieve a Public Key Confirmation from the HSM.

> **NOTE** This confirmation procedure is currently not supported on FM-enabled HSMs. Refer to FM Deployment Constraints for details.

Syntax

cmu getpkc [{**-handle**=<handle#> | **-ouid**=<OUID#>}] [**-outputfile**=<filename>] [**-verify**]

Argument(s)	Description
-handle=<handle#>	The handle to the corresponding private key for the PKC. This method of selection applies to Luna HSMs only. On a Luna Cloud HSM service slot, use **-ouid**.
-ouid=<OUID#>	The Object Unified Identifier (OUID) to the corresponding private key for the PKC. This method of selection requires Luna HSM Client 10.2.0 or newer, and applies to Luna Cloud HSM services only. On a Luna HSM slot, use **-handle**.
-outputfile=<filename>	The name of the file that receives the PKC.
-verify	Sets a flag to verify the PKC against the certificate that signed the PKC. It must be set to True or False (or 1 or 0), with False being the default.

If you run the command with no parameters, you are prompted for the mandatory ones.

See also "Common CMU Options" on page 10.

Example

```
cmu getpkc -handle=5
```

cmu import

This function:

- Imports X.509 certificates from a file to the token or HSM. The file may include a single DER encoded binary certificate or a CMSS PKCS #7 certificate or certificate set. Either type of certificate can be binary or PEM (base 64) encoded. An optional label can be defined as a function parameter. If omitted, the common name of the certificate subject is chosen as the label.
- Imports a public key onto an HSM partition

Syntax

cmu import -inputFile=<filename> [**-label**=<label>] [**-pubkey**=<keytype>]

Argument(s)	Description
-inputFile=<filename>	Defines the name of the file containing the certificate to import.
-label=<label>	Defines a label to apply to the imported file. If the file is a certificate, and no label is defined, the Common Name portion of the certificate distinguished name is used instead. If the file is a public key, it can be any text you care to apply.
-private=<T> or <F>	Defines whether a certificate is created in the private space (default is -private=T). Set -private=F to make the created certificate publicly accessible for applications that need to acquire the certificate without need for authentication.
-pubkey=<keytype>	When the input file is a public key, defines the type of key to be imported. Use lowercase.

See also "Common CMU Options" on page 10.

Example

The following example inputs the public key in **secp521r1-pub.pem**

```
cmu import –in secp521r1-pub.pem –label ID3pubkey –pubkey=ecdsa
Select token
  [0] Token Label: tsb012
  [1] Token Label: txb161
  Enter choice: 1
Please enter password for token in slot 1 : *******

cmu list
Select token
  [0] Token Label: tsb012
  [1] Token Label: txb161
  Enter choice: 1
Please enter password for token in slot 1 : *******
handle=235       label=ID3pubkey
```

cmu importkey

This function unwraps an RSA, DSA , or ECDSA private key onto the selected token or HSM. The key file may be in any of the following formats:

- PKCS #12(PFX) RSA in a DER-encoded format (.pfx file)
- PKCS #8(Unencrypted PrivatekeyInfo) in RSA or DSA in base 64 PEM, or binary DER format
- PKCS #1 (RSA in base64 PEM, or binary DER) format
- ECDSA keys can be PKCS1, PKCS8, and PKCS12 format.

> **NOTE** PKCS#12 encrypted keys can be imported into the HSM from Luna HSM Firmware 7.7.0 and Luna HSM Client 10.3.0 and newer.

Syntax

cmu importkey -in=<filename> **-keyalg**=<algorithm> [**-wrapkey**=<handle/OUID>] [**-setkeyattr**] [**-PKCS8**] [**-PKCS12**]

Argument(s)	Description
-in=<filename>	Defines the full path to the file containing the PEM- or DER-encoded key to import.
-keyalg=<algorithm>	Specifies the key's algorithm. **Valid values:** DSA,RSA,ECDSA
-out=<filename>	Defines the full path to the file containing the PEM- or DER-encoded key to import.
-PKCS8	Indicates that the key to import is formatted according to the PKCS#8 standard. **NOTE: cmu** options are case-sensitive.
-PKCS12	Indicates that the key to import is formatted according to the PKCS#12 standard. Only the private key portion is unwrapped onto the token. Any certificates in this file are simply ignored. It is assumed that you properly export a PKCS #12 key from Windows keystore (or other source, as appropriate). **NOTE: cmu** options are case-sensitive.
-setkeyattr	Allows the user to manually enter the imported key's attributes. Modifiable key attributes are CKA_DECRYPT, CKA_SIGN, CKA_EXTRACTABLE, and CKA_UNWRAP. The defaults are always 1=true.

Argument(s)	Description
-wrapkey=<handle/OUID>	The handle or OUID of the existing key that is to be used as the wrapping key. This key must have the CKA_WRAP attribute set to true. If this flag is not specified the default behavior is to auto-generate an AES key for the sole purpose of unwrapping the key onto the HSM. **NOTE** The OUID can be specified on a Luna Cloud HSM service slot only, and requires Luna HSM Client 10.2.0 or newer. On a Luna HSM slot, specify the key by its object handle.

See also "Common CMU Options" on page 10.

Example

```
cmu importkey -in rawrsa1028.pem -keyalg RSA -wrapkey 11 -setkeyattr

cmu importkey -PKCS8 -in pk8privkey.pem -keyalg DSA

cmu importkey -in rsakey.pem -keyalg RSA -wrapkey 11

cmu importkey -in rsakey.pem -keyalg RSA

cmu importkey -PKCS12 -in p12.pfx -keyalg RSA

cmu importkey -PKCS12 -in ec.pfx -keyalg ECDSA
```

NOTE PKCS#12 encrypted keys can be imported into the HSM *from Luna HSM Firmware 7.7.0 and Luna HSM Client 10.3.0 and newer in non-FIPS mode* (HSM policy 12 set to ON).

NOTE

1. Ideally, the private key should be in PKCS#8 format (privatekeyinfo) and not encrypted.
   To convert a private key of either RSA or DSA type: (see PKCS#1 for RSA and PKCS#11 (11.9) for DSA) into a PKCS#8 structure, use the following openssl command
   openssl pkcs8 -in key.pem –nocrypt -topk8 -out noenckey.pem
   You are prompted for the password to decrypt the PrivateKeyInfo.
2. If the PKCS#8 structure is already encrypted according to the PKCS#5-PBE standard, then to import via CMU, use the following command
   openssl pkcs8 -in pk8.pem -out key.pem
   You are prompted for the password to decrypt the PrivateKeyInfo.
3. You can export the PrivatekeyInfo contents of a .pfx file by using the following openssl command
   openssl pkcs12 –in p12.pfx –out pk12_privkey.pem –nocerts –nodes
   You are prompted for the password to decrypt the PrivateKeyInfo.

cmu list

This function lists all objects (keys, certificates and other general data objects) on the HSM that match an optional set of search criteria and that are accessible given the authentication state of the HSM. Search criteria can include many of the object attributes that are available for searching via the PKCS #11 API. If no search criteria are defined, all accessible objects are returned. The content of the entries in the returned list is definable and can include any combination of viewable object attributes. The default is to include the handle (on Luna partitions) or OUID (on Luna Cloud HSM) and the label (CKA_LABEL).

Syntax

cmu list [**-display**=<attributes>] [**-class**=<class>] [**-keyType**=<type>] [**-certificateType**=<type>] [**-label**=<label>] [**-application**=<attribute>] [**-value**=<value>] [**-issuer**=<issuer>] [**-serialNumber**=<SN>] [**-subject**=<subject>] [**-id**=<ID>] [**-token**=<0/1>] [**-modulusBits**=<length>] [**-publicExponent**=<value>] [**-private**=<0/1>] [**-sensitive**=<0/1>] [**-alwaysSensitive**=<0/1>] [**-extractable**=<0/1>] [**-neverExtractable**=<0/1>] [**-local**=<0/1>] [**-encrypt**=<0/1>] [**-decrypt**=<0/1>] [**-sign**=<0/1>] [**-verify** <0/1>] [**-wrap** <0/1>] [**-unwrap** <0/1>] [**-derive**=<0/1>] [**-startDate**=<YYYYMMDD>] [**-endDate**=<YYYYMMDD>] [**-modifiable**=<0/1>]

Argument(s)	Description
-alwaysSensitive=<0/1>	Show objects that match value True or False (or 1 or 0).
-application=<attribute>	Specifies the application attribute that objects must match in order to be listed.
-certificateType=<type>	Specifies the type of certificate to list. It can only be set to x.509 if used. **Valid values:** x.509
-class=<class>	Specifies the class of object to list. **Valid values:** data,certificate,public,private,secret
-decrypt=<0/1>	Show objects that match value True or False (or 1 or 0).
-derive=<0/1>	Show objects that match value True or False (or 1 or 0).
-display=<attributes>	Specifies the attributes to be displayed for each returned object in the list. Multiple attributes can also be specified by repeated use of the display option instead of using the comma-separated list. If this parameter is omitted, only the handle/OUID and label are displayed. **Valid values:** index,handle,ouid,class,keyType,label,value **NOTE** OUID can be specified on a Luna Cloud HSM service slot only, and requires Luna HSM Client 10.2.0 or newer. On a Luna HSM slot, object handles are displayed.
-encrypt=<0/1>	Show objects that match value True or False (or 1 or 0).
-endDate=<YYYYMMDD>	This option specifies the end date that objects must match in order to be listed.

Argument(s)	Description
-extractable=<0/1>	Show objects that match value True or False (or 1 or 0).
-id=<ID>	Specifies the ID that objects must match in order to be listed.
-issuer=<issuer>	Specifies the issuer that objects must match in order to be listed.
-keyType=<type>	Specifies the type of keys to list. **Valid values:** rsa,dsa,dh,des,2des,3des,rc2,rc4,rc5,cast3,cast5,generic
-label=<label>	Specifies the label that objects must match in order to be listed.
-local=<0/1>	Show objects that match value True or False (or 1 or 0).
-modifiable=<0/1>	Show objects that match value True or False (or 1 or 0).
-modulusBits=<length>	This option specifies the modulus size that RSA keys must match in order to be listed.
-neverExtractable=<0/1>	Show objects that match value True or False (or 1 or 0).
-private=<0/1>	Show objects that match value True or False (or 1 or 0).
-publicExponent=<value>	This option specifies the public exponent value that RSA keys must match in order to be listed. It can only be set to 3, 17 or 65537. Only 65537 is allowed in FIPS mode.
-sensitive=<0/1>	Show objects that match value True or False (or 1 or 0).
-serialNumber=<SN>	Specifies the serial number that objects must match in order to be listed.
-sign=<0/1>	Show objects that match value True or False (or 1 or 0).
-startDate=<YYYYMMDD>	This option specifies the start date that objects must match in order to be listed.
-subject=<subject>	Specifies the subject that objects must match in order to be listed.
-token=<0/1>	Specifies whether permanent or temporary objects are to be listed. **Valid values: 0**(temporary objects),**1**(permanent objects)
-unwrap=<0/1>	Show objects that match value True or False (or 1 or 0).
-value=<value>	Specifies the value that objects must match in order to be listed.
-verify=<0/1>	Show objects that match value True or False (or 1 or 0).
-wrap=<0/1>	Show objects that match value True or False (or 1 or 0).

See also "Common CMU Options" on page 10.

Example

The following example displays the handle and label of each certificate that is accessible on the HSM:

```
cmu list -class=certificate
```

The following example displays the handles of all locally generated RSA private signing keys on the HSM:

```
cmu list -keyType=rsa -local=True -sign=True -display=handle
```

The following example displays the class, type and label of all signing keys on the HSM:

```
cmu list -display=class,keyType,label -sign=True
```


cmu requestcertificate

This function creates a PKCS #10 certificate request for an RSA/DSA/ECDSA key pair on the token or HSM. It must be provided with the handle/OUID either to the public key or to the corresponding private key (all of the public key components are contained within the private key). The private key must have signing capability because it is used to sign the certificate request structure. The signature is done using any of the mechanisms listed below. The subject name is defined by a series of optional RDN components.

If none of these components are provided on the command line, the CKA_SUBJECT of the private key is used as the subject of the certificate request. If the private key does not have its CKA_SUBJECT attribute set, the user will be queried for each of the RDN components. The Subject DN should contain at least the country, organization and common name components.

The signed certificate request is output to the specified file.

Syntax

cmu requestCertificate {**-publichandle**=<pubkeyhandle#> | **-publicouid**=<pubkeyOUID#>} {**-privatehandle**=<privkeyhandle#> | **-privateouid**=<privkeyOUID#>} **-outputFile**=<filename> [**-sha1WithRsa**] [**-sha224withrsa**] [**-sha256withrsa**] [**-sha384withrsa**] [**-sha512withrsa**] [**-sha1withdsa**] [**-sha1withecdsa**] [**-sha224withecdsa**] [**-sha256withecdsa**] [**-sha384withecdsa**] [**-sha512withecdsa**] [**-C**=<country>] [**-S**=<state>] [**-L**=<locality>] [**-O**=<organization>] [**-OU**=<org_unit>] [**-CN**=<common_name>] [**-e**=<e-mail_address>] [**-binary**]

Argument(s)	Description
-binary	Defines the certificate request format to be raw binary (DER encoding) instead of the default PEM (base64) encoding.
-C=<country>	Defines the two-letter country name for the subject distinguished name (DN) of the certificate request. This parameter should be present in the subject DN.
-CN=<common_name>	Defines the common name for the subject distinguished name (DN) of the certificate request. This parameter should be present in the subject DN.
-E=<e-mail_address>	Official or contact e-mail address of certificate authority.
-L=<locality>	Defines the locality (typically the city) for the subject distinguished name of the certificate request. This parameter may be present in the Subject DN.
-md5withrsa	Defines the signature algorithm for the certificate request to be pkcs-1-md5withRSAEncryption. The default is to use sha1WithRsa.
-multiorg	For Organization Name and Organization Unit name, the user may make multiple entries if the **-multiorg** option was provided. If you are using a Luna Network HSM with Luna HSM Firmware 7.3.0 and newer and Luna HSM Client 7.3.0 and newer, the **-multiorg** option can only be specified when creating a certificate request interactively; that is, by inputting all arguments as responses to prompts.

Argument(s)	Description
-O=<organization>	Defines the organization name for the subject distinguished name (DN) of the certificate request. This parameter should be present in the subject DN.
-OU=<org_unit>	Defines the organization unit name for the subject distinguished name (DN) of the certificate request. This parameter may be present in the subject DN.
-outputFile=<filename>	Defines the file that receives the certificate request.
-privatehandle =<privkeyhandle#>	Defines the handle to the private key from an RSA key pair to be certified. If this parameter is omitted and there is only one private signing key on the partition, that key is automatically selected. If this parameter is omitted and there are multiple private signing keys on the partition, the user is asked to select the private signing key. This method of selection applies to Luna HSMs only. On a Luna Cloud HSM service slot, use **-privateouid**.
-privateouid=<privkeyOUID#>	Defines the Object Unified Identifier (OUID) of the private key from an RSA key pair to be certified. If this parameter is omitted and there is only one private signing key on the partition, that key is automatically selected. If this parameter is omitted and there are multiple private signing keys on the partition, the user is asked to select the private signing key. This method of selection requires Luna HSM Client 10.2.0 or newer, and applies to Luna Cloud HSM services only. On a Luna HSM slot, use **-privatehandle**.
-publichandle =<pubkeyhandle#>	Defines the handle to the public key from an RSA key pair to be certified. If this parameter is omitted and there is only one public signing key on the HSM, that key is automatically selected. If this parameter is omitted and there are multiple public signing keys on the HSM, the user is asked to select the public signing key. This method of selection applies to Luna HSMs only. On a Luna Cloud HSM service slot, use **-publicouid**.
-publicouid=<pubkeyOUID#>	Defines the Object Unified Identifier (OUID) of the public key from an RSA key pair to be certified. If this parameter is omitted and there is only one public signing key on the partition, that key is automatically selected. If this parameter is omitted and there are multiple public signing keys on the partition, the user is asked to select the public signing key. This method of selection requires Luna HSM Client 10.2.0 or newer, and applies to Luna Cloud HSM services only. On a Luna HSM slot, use **-publichandle**.
-S=<state>	Defines the state or province name for the subject distinguished name of the certificate request. This parameter may be present in the Subject DN.
-sha1withdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha1withDSAEncryption. The default is to use sha1WithRsa.
-sha1withecdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha1withECDSAEncryption. The default is to use sha1WithRsa.

Argument(s)	Description
-sha1WithRsa	Defines the signature algorithm for the certificate request to be pkcs-1-SHA1withRSAEncryption. The default is to use sha1WithRsa.
-sha224withecdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha224withECDSAEncryption. The default is to use sha1WithRsa.
-sha224withrsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha224withRSAEncryption. The default is to use sha1WithRsa.
-sha256withecdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha256withECDSAEncryption. The default is to use sha1WithRsa.
-sha256withrsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha256withRSAEncryption. The default is to use sha1WithRsa.
-sha384withecdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha384withECDSAEncryption. The default is to use sha1WithRsa.
-sha384withrsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha384withRSAEncryption. The default is to use sha1WithRsa.
-sha512withecdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha512withECDSAEncryption. The default is to use sha1WithRsa.
-sha512withrsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha512withRSAEncryption. The default is to use sha1WithRsa.

See also "Common CMU Options" on page 10.

> **TIP** When requesting a certificate (**cmu requestcertificate**) using the wrong attribute to specify the private key, an incorrect error message is thrown ("**Signing key not found**"). Instead, use **-privatehandle** to specify a key on a Luna partition, and **-privateouid** on a Luna Cloud HSM service.

Examples

The following example supports Luna HSM Client 10.1.0 and older:

```
cmu requestCert -privatehandle=7 -publichandle=6 -C=CA -L=Ottawa -O=Thales -CN=TestCertificate
-outputFile=testCert.req
```

The following example supports Luna HSM Client 10.2.0 and newer with Luna Cloud HSM service slots:

```
cmu requestCert -privateouid=650200000d0000b8397e0800 -publicouid=640200000d0000b8397e0800 -
C=CA -L=Ottawa -O=Thales -CN=TestCertificate -outputFile=testCert.req
```

cmu selfsigncertificate

This function creates a self-signed X.509 certificate for an RSA, DSA, or ECDSA key pair on the partition. It must be provided with the handles/OUIDs to both the public key and the corresponding private key (all of the public key components are contained within the private key). The private key must have Signing capability since it is used to sign the certificate request structure. The signature is done with any of the mechanisms listed below. The subject name is defined by a series of optional RDN components.

If none of these components are provided on the command line, the CKA_SUBJECT of the private key is used as the subject of the certificate. If the private key does not have its CKA_SUBJECT attribute set, the user will be queried for each of the RDN components. The Subject DN should contain at least the country, organization and common name components.

The certificate will, by default, have a keyUsage setting of keycertsign. The certificate is stored as a PKCS #11 certificate object on the token. The CKA_ID attribute of the certificate is defined by an optional parameter. If this parameter is omitted, the CKA_ID of the private key is used.

Syntax

cmu selfSignCertificate {**-publichandle**=<pubkeyhandle#> | **-publicouid**=<pubkeyOUID#>} {**-privatehandle**=<privkeyhandle#> | **-privateouid**=<privkeyOUID#>} **-private=<T/F> -serialNumber**=<SN> **-startDate**=<YYYYMMDD> **-endDate**=<YYYYMMDD> [**-label**=<label>] [**-id**=<CKA_ID>] [**-keyids**=<value>] [**-keyidalg**=<algorithm>] [**-keyusage**=<type(s)>] [**-md5WithRsa**] [**-sha1WithRsa**] [**-sha224withrsa**] [**-sha256withrsa**] [**-sha384withrsa**] [**-sha512withrsa**] [**-C**=<country>] [**-S**=<state>] [**-L**=<locality>] [**-O**=<organization>] [**-OU**=<org_unit>] [**-CN**=<common_name>]

Argument(s)	Description
-basicconstraints =<constraints>	Defines constraints applied to the certificate. Can include one or more in a comma-delimited list. **Valid Values:** critical,optional,ca:true,ca:false,pathlen:[value < 127]
-C=<country>	Defines the two-letter country name for the subject distinguished name (DN) and issuer Distinguished Name of the certificate. This parameter should be present in each DN.
-CN=<common_name>	Defines the common name for the subject DN and issuer DN of the certificate. This parameter SHOULD be present in each DN.
-endDate <YYYYMMDD>	Defines the validity end of the certificate, in the format YYYYMMDD.
-extendedkeyusage =<usages>	Defines the permitted additional usage of the key. Can include one or more in a comma-delimited list. **Valid Values:** critical,optional,clientauth,serverauth,codesigning,emailprotection,timestamping,ocspsigningD

Argument(s)	Description
-id=<CKA_ID>	Defines the CKA_ID attribute for the certificate object that gets created on the HSM. If omitted, the CKA_ID attribute of the private key is used instead.
-keyidalg =<algorithm>	Specifies the hashing algorithm used to create the subject key identifier (SKI) and authority key identifier (AKI) of the newly created certificate. This option is used with **-keyids**. **Valid values:** **> sha1** **> sha224** **> sha256** **> sha384** **> sha512** **NOTE** This parameter is only available if you are using a Luna Network HSM with Luna HSM Client 10.3.0 and newer.
-keyids =<value>	Indicates whether the newly created certificate will have an SKI and AKI. The SKI is created using the hashing algorithm specified with **-keyidalg**. If no algorithm is specified with **-keyidalg**, the SKI and AKI are created using SHA-1. **Valid values:** 1,0 (True or False) **NOTE** This parameter is only available if you are using a Luna Network HSM with Luna HSM Client 10.3.0 and newer.
-keyusage =<type(s)>	Defines the key usage extension for the certificate. This parameter may be included more than once to define multiple usages, or it can be used once with a comma-separated list of usage types. If no key usage is specified, a default setting of keycertsign is used. **Valid values:** digitalsignature,nonrepudiation,keyencipherment,dataencipherment,keyagreement,keycertsign,crlsign,encipheronly,decipheronly.
-L=<locality>	Defines the locality (typically the city) for the subject DN and issuer DN of the certificate. This parameter MAY be present in each DN.
-label=<label>	Defines the CKA_LABEL attribute for the certificate object that gets created on the HSM. If omitted, the common name of the issuer and subject DN is used instead.
-md5WithRsa	Defines the signature algorithm for the certificate request to be pkcs-1-MD5withRSAEncryption. The default is to use sha1WithRsa.
-multiorg	For Organization Name and Organization Unit name, the user may make multiple entries if the **-multiorg** option was provided.

Argument(s)	Description
-O =<organization>	Defines the organization name for the subject DN and issuer DN of the certificate. This parameter SHOULD be present in each DN.
-OU=<org_unit>	Defines the organization unit name for the subject DN and issuer DN of the certificate. This parameter MAY be present in each DN.
-private=<T/F>	Defines whether a certificate is created in the private space (default is F). Set **-private=T** to require authentication before applications can use the certificate.
-privatehandle =<privkeyhandle#>	Defines the handle to the private key from an RSA key pair to be certified. If this parameter is omitted and there is only one private signing key on the HSM, that key is automatically selected. If this parameter is omitted and there are multiple private signing keys on the HSM, the user is asked to select the private signing key. This method of selection applies to Luna HSMs only. On a Luna Cloud HSM service slot, use **-publicouid**.
-privateouid =<privkeyOUID#>	Defines the Object Unified Identifier (OUID) of the private key from an RSA key pair to be certified. If this parameter is omitted and there is only one private signing key on the partition, that key is automatically selected. If this parameter is omitted and there are multiple private signing keys on the partition, the user is asked to select the private signing key. This method of selection requires Luna HSM Client 10.2.0 or newer, and applies to Luna Cloud HSM services only. On a Luna HSM slot, use **-privatehandle**.
-publichandle =<pubkeyhandle#>	Defines the handle to the public key from an RSA key pair to be certified. If this parameter is omitted and there is only one public signing key on the HSM, that key is automatically selected. If this parameter is omitted and there are multiple public signing keys on the HSM, the user is asked to select the public signing key. This method of selection applies to Luna HSMs only. On a Luna Cloud HSM service slot, use **-publicouid**.
-publicouid =<pubkeyOUID#>	Defines the Object Unified Identifier (OUID) of the public key from an RSA key pair to be certified. If this parameter is omitted and there is only one public signing key on the partition, that key is automatically selected. If this parameter is omitted and there are multiple public signing keys on the partition, the user is asked to select the public signing key. This method of selection requires Luna HSM Client 10.2.0 or newer, and applies to Luna Cloud HSM services only. On a Luna HSM slot, use **-publichandle**.
-S=<state>	Defines the state or province name for the subject DN and issuer DN of the certificate. This parameter may be present in each DN.
-serialNumber =<SN>	Defines the serial number of the certificate, in big-endian hexadecimal form.
-sha1withdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha1withDSAEncryption. The default is to use sha1WithRsa.

Argument(s)	Description
-sha1withecdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha1withECDSAEncryption. The default is to use sha1WithRsa.
-sha1WithRsa	Defines the signature algorithm for the certificate request to be pkcs-1-SHA1withRSAEncryption. The default is to use sha1WithRsa.
-sha224withecdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha224withECDSAEncryption. The default is to use sha1WithRsa.
-sha224withrsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha224withRSAEncryption. The default is to use sha1WithRsa.
-sha256withecdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha256withECDSAEncryption. The default is to use sha1WithRsa.
-sha256withrsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha256withRSAEncryption. The default is to use sha1WithRsa.
-sha384withecdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha384withECDSAEncryption. The default is to use sha1WithRsa.
-sha384withrsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha384withRSAEncryption. The default is to use sha1WithRsa.
-sha512withecdsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha512withECDSAEncryption. The default is to use sha1WithRsa.
-sha512withrsa	Defines the signature algorithm for the certificate request to be pkcs-1-sha512withRSAEncryption. The default is to use sha1WithRsa.
-startDate =<YYYYMMDD>	Defines the validity start of the certificate, in the format YYYYMMDD.

See also "Common CMU Options" on page 10.

Example

The following example creates a self-signed certificate for RSA key 3161181396:

```
# cmu selfsigncertificate -slot 6 -password userpin -publichandle 3161181396  -privatehandle
1196747189 -serialNum 0133337f -C CA -S ON -L Ottawa -O SafeNet -OU PD -CN test_cmu_cert -
startDate 20120920 -endDate 20220920
```

```
Certificate Management Utility (64-bit) v10.1.0-32. Copyright (c) 2019 SafeNet. All
rights reserved.

Using "CKM_SHA256_RSA_PKCS" Mechanism
```

cmu setattribute

This function sets any modifiable attributes for an object. An optional input filename can be used to specify a file from which the new attribute values are to be read.

Syntax

cmu setAttribute {**-handle**=<handle#> | **-ouid**=<OUID#>} [**-inputFile**=<filename>] [**-label**=<label>] [**-application**=<value>] [**-value**=<value>] [**-issuer**=<issuer>] [**-serialNumber**=<SN>] [**-subject**=<subject>] [**-id**=<hex_ID>] [**-extractable**=<0>] [**-startDate**=<YYYYMMDD>] [**-endDate**=<YYYYMMDD>] [**-extractable**=<0>] [**-encrypt**=<0/1>] [**-decrypt**=<0/1>] [**-sign**=<0/1>] [**-verify**=<0/1>] [**-wrap**=<0/1>] [**-unwrap**=<0/1>] [**-derive**=<0/1>] [**-sensitive**=<0/1>]

Argument(s)	Description
-application =<value>	Defines a new value for the application attribute of a data object on the HSM.
-decrypt=<0/1>	Set to True or False (or 1 or 0). Note that an HSM is typically configured such that functional key attributes cannot be changed, so attempting to change this attribute will be rejected by the HSM.
-derive=<0/1>	Set to True or False (or 1 or 0). Note that an HSM is typically configured such that functional key attributes cannot be changed, so attempting to change this attribute will be rejected by the HSM.
-encrypt=<0/1>	Set to True or False (or 1 or 0). Note that an HSM is typically configured such that functional key attributes cannot be changed, so attempting to change this attribute will be rejected by the HSM.
-endDate=<YYYYMMDD>	Defines a new endDate field for a key on the HSM. The format for the value is YYYYMMDD.
-extractable=<0>	Defines a new extractable setting for a private key on the HSM. This setting can only be changed from True to False (or from 1 to 0).
-handle=<handle#>	Defines the handle of the object. If this parameter is omitted and there is only one object on the partition, that object is automatically selected. If this parameter is omitted and there are multiple objects on the partition, the user is asked to select the object. This method of selection applies to Luna HSMs only. On a Luna Cloud HSM service slot, use **-ouid**.
-id=<hex_ID>	Defines a new ID field for a key or certificate on the HSM. It must be set to a big-endian hexadecimal integer value.
-inputFile=<filename>	Names a file from which to obtain additional attribute settings. The settings in this file shall be one per line and of the form: <attributeName>**=**<attributeValue>

Argument(s)	Description
-issuer=<issuer>	Defines a new issuer attribute for a certificate on the HSM. It must be set to a big-endian hexadecimal integer value. Note that this field is informational, typically set to the DER encoding of the issuer field within the certificate, and changing it does not affect the actual issuer field within the certificate itself.
-label=<label>	Defines a new label of an object on the HSM. If this parameter is omitted and there is only one object on the partition, that object is automatically selected. If this parameter is omitted and there are multiple objects on the partition, the user is asked to select the object.
-ouid=<OUID#>	Defines the Object Unified Identifier (OUID) of the object. If this parameter is omitted and there is only one object on the partition, that object is automatically selected. If this parameter is omitted and there are multiple objects on the partition, the user is asked to select the object. This method of selection requires Luna HSM Client 10.2.0 or newer, and applies to Luna Cloud HSM services only. On a Luna HSM slot, use **-handle**.
-sensitive=<0/1>	Set to True or False (or 1 or 0). Note that an HSM is typically configured such that functional key attributes cannot be changed, so attempting to change this attribute will be rejected by the HSM.
-serialNumber=<SN>	Defines a new serial number attribute for a certificate on the HSM. It must be set to a big-endian hexadecimal integer value. Note that this field is informational, typically set to the DER encoding of the serial number of the certificate, and changing it does not affect the actual serial number field within the certificate itself.
-sign=<0/1>	Set to True or False (or 1 or 0). Note that an HSM is typically configured such that functional key attributes cannot be changed, so attempting to change this attribute will be rejected by the HSM.
-startDate=<YYYYMMDD>	Defines a new startDate field for a key on the HSM. The format for the value is YYYYMMDD.
-subject=<subject>	Defines a new subject field for an object on the HSM. It must be set to a big-endian hexadecimal integer value. The subject field is typically set to the DER encoding of the subject distinguished name for the key or certificate. Note that the subject is not modifiable for certificate objects once they are created.
-unwrap=<0/1>	Set to True or False (or 1 or 0). Note that an HSM is typically configured such that functional key attributes cannot be changed, so attempting to change this attribute will be rejected by the HSM.
-value=<value>	Defines a new value attribute for an object on the HSM. It must be set to a big-endian hexadecimal integer value. Note that the value attribute can be changed only for data objects, not for certificates or keys.

Argument(s)	Description
-verify=<0/1>	Set to True or False (or 1 or 0). Note that an HSM is typically configured such that functional key attributes cannot be changed, so attempting to change this attribute will be rejected by the HSM.
-wrap=<0/1>	Set to True or False (or 1 or 0). Note that an HSM is typically configured such that functional key attributes cannot be changed, so attempting to change this attribute will be rejected by the HSM.

See also "Common CMU Options" on page 10.

Example

The following example changes the key with handle 43 to be unextractable:

```
cmu setAttribute -handle=43 -extractable=False
```

cmu verifyhsm

This command allows you to verify that the client is connected to a genuine Luna HSM, by creating and verifying a confirmation on a temporary key created in the HSM. It also includes a proof of possession that asks the HSM to sign a user-entered string as proof the associated private key is present within the target HSM.

The root certificate is not included in the client package. Download the safenet-root.pem and copy it to your client directory to execute **cmu verifyhsm**.

Refer also to Verifying the HSM's Authenticity.

> **NOTE** This confirmation procedure is currently not supported on FM-enabled HSMs. Refer to FM Deployment Constraints for details.

Syntax

cmu verifyhsm -challenge="<string>**"** [**-rootcert**=<filename>]

Argument(s)	Description
-challenge=<string>	Defines a user-entered string for the HSM to sign.
-rootcert=<filename>	Defines the name of the .pem file that contains the root certificate.

See also "Common CMU Options" on page 10.

Example

```
./cmu verifyhsm -challenge "1234567890" -rootcert safenet-root.pem
Select token
 [0] Token Label: mypartition-1
 [1] Token Label: mypartition-2
 Enter choice: 0
Please enter password for token in slot 0 : *******
Reading rootcert from file "safenet-root.pem"... ok.
Generating temporary RSA keypair in HSM... ok.
Extracting PKC bundle from HSM... ok.
Verifying PKC certificate... ok.
Verifying DAC certificate... ok.
Verifying HOC certificate... ok.
Verifying MIC certificate... ok.
Verifying MIC against rootcert... ok.
Signing and verifying challenge... ok.
Verifying HSM serial number... ok.
Overall status: Success.
```

cmu verifypkc

Verify a Public Key Confirmation from the HSM.

Syntax

cmu verifypkc -inputFile=<filename>

Argument(s)	Description
-inputFile =<filename>	Defines the name of the file that contains the PKC.

See also "Common CMU Options" on page 10.

Example

```
cmu verifypkc –inputFile=test.pkc
```

CHAPTER 2: ckdemo

> **NOTE** This is a general-purpose tool intended for use across Luna HSM versions. It might reference mechanisms and features that are not available on all Luna products.

The **ckdemo** utility is a simple console-based tool that provides a menu of functions that perform operations based on the PKCS#11 API. The options/operations are generally low-level, atomic commands, that would need to be combined to perform useful actions. The purpose of ckdemo is to allow you to become familiar with the low-level building-block commands and combinations that you would then program into your application, using the Software Development Kit and API.

Accessing ckdemo

The **ckdemo** utility is included with the Luna HSM Client.

> **NOTE** As a general rule, you would need to
> 1. open a session (option 1)
> 2. log in (option 3)
>
> before using further ckdemo options.

To access ckdemo from a Linux client:

1. Go to the Luna HSM Client binary directory.

   cd /usr/safenet/lunaclient/bin

2. Launch the ckdemo utility.

   ./ckdemo

To access ckdemo from a Windows client:

1. Navigate to the Luna HSM Client installation folder (**C:\Program Files\SafeNet\LunaClient**).
2. Double-click on **ckdemo** to open a console window with the ckdemo interface.

Using the Menu

When you launch the **ckdemo** utility, the menu provides access to functions organized by category.

To execute functions listed in the menu, type the number of the function and press **Enter**. You will be prompted to provide additional parameters as required. Since most commands represent multiple HSM functions, you may need to use more than one command to accomplish a task. For example, many commands require that you first open a session on a token slot or HSM partition (function **1**). Others require that you first login to the HSM or partition (function **3**).

Authentication or initialization functions may require the Luna PED. If the Luna PED is connected and ready when a command is issued, it prompts the user for the appropriate action. Otherwise, the command times out. If you do not provide the requested PED key or keypad input, the Luna PED times out and returns an error to the calling application (in this case, **ckdemo**).

The **ckdemo** functions are described in the following sections:

> "AUDIT/LOG Menu Functions" on the next page
> "CA Menu Functions" on page 48
> "CLUSTER EXECUTION Menu Functions" on page 50
> "HIGH AVAILABILITY RECOVERY Menu Functions" on page 50
> "KEY Menu Functions" on page 51
> "OBJECT MANAGEMENT Menu Functions" on page 52
> "OFFBOARD KEY STORAGE Menu Functions" on page 54
> "OTHERS Menu Functions" on page 54
> "PED INFO Menu Functions" on page 58
> "POLICY Menu Functions" on page 58
> "SCRIPT EXECUTION Menu Functions" on page 59
> "SECURITY Menu Functions" on page 59
> "SRK Menu Functions" on page 60
> "TOKEN Menu Functions" on page 60
> "KEY AUTHORIZATION Menu Functions" on page 52

Example

```
TOKEN:
    ( 1) Open Session  ( 2) Close Session  ( 3) Login
    ( 4) Logout        ( 5) Change PIN     ( 6) Init Token
    ( 7) Init Pin      ( 8) Mechanism List ( 9) Mechanism Info
    (10) Get Info      (11) Slot Info      (12) Token Info
    (13) Session Info  (14) Get Slot List  (15) Wait for Slot Event
    (16) Token Status  (18) Factory Reset  (19) CloneMofN
    (33) Token Insert  (34) Token Delete
    (36) Show Roles    (37) Show Role Configuration Policies
    (38) Show Role State   (39) Get OUID
    (58) HSM Zeroize       (59) Token Zeroize
OBJECT MANAGEMENT:
    (20) Create object (21) Copy object    (22) Destroy object
    (23) Object size   (24) Get attribute  (25) Set attribute
                       (26) Find object    (27) Display Object
    (30) Modify Usage Count         (31) Destroy Multiple Objects
    (32) Extract Public Key
SECURITY:
    (40)  Encrypt file (41) Decrypt file   (42)  Sign
    (43)  Verify       (44) Hash file      (45)  Simple Generate Key
                                           (46)  Digest Key
HIGH AVAILABILITY RECOVERY:
    (50) HA Init       (51) HA Login     (52) HA Status
```

```
KEY:
    (60) Wrap key      (61) Unwrap key     (62) Generate random number
    (63) Derive Key    (64) PBE Key Gen    (65) Create known keys
    (66) Seed RNG      (67) EC User Defined Curves
CA:
    (70) Set Domain    (71) Clone Key      (72) Set MofN
    (73) Generate MofN (74) Activate MofN  (75) Generate Token Keys
    (76) Get Token Cert Info               (77) Sign Token Cert
    (78) Generate CertCo Cert              (79) Modify MofN
    (86) Dup. MofN Keys                    (87) Deactivate MofN
    (88) Get Token Certificates            (112) Set Legacy Cloning Domain
OTHERS:
    (90) Self Test
    (94) Open Access    (95) Close Access
    (97) Set App ID     (98) Options
OFFBOARD KEY STORAGE:
   (101) Extract Masked Object            (102) Insert Masked Object
   (103) Multisign With Value             (104) Clone Object
   (105) SIMExtract                       (106) SIMInsert
   (107) SimMultiSign                     (118) Extract Object
                                          (119) Insert Object
SCRIPT EXECUTION:
   (108) Execute Script                   (109) Execute Asynchronous Script
                                          (110) Execute Single Part Script
CLUSTER EXECUTION:
   (111) Get Cluster State
   (113) Lock Clustered Slot              (114) Unlock Clustered Slot
PED INFO:
   (120) Set Ped Info   (121) Get Ped Info (122) Init RPV
   (123) Delete RPV
AUDIT/LOG:
   (130) Get Config     (131) Set Config   (132) Verify logs
   (133) Get Time       (134) Set Time     (135) Import Secret
   (136) Export Secret  (137) Init Audit   (138) Get Status
   (139) Log External
SRK:
   (200) SRK Get State  (201) SRK Restore  (202) SRK Resplit
   (203) SRK Zeroize    (204) SRK Enable/Disable

KEY AUTHORIZATION
   (210) Authorize Key               (211) Set Authorization Data
   (212) ReSet Authorization Data    (213) Assign Key

POLICY:
   (53) Show Partition Policies     (54) Set Partition Policies
   (55) Show HSM Policies (56) Set HSM Policies (57) Set Destructive HSM Policies

(TITLE) menu titles, (99 or FULL) Full Help, (NONE) No help, (0 or EXIT) Quit

Enter your choice :
```

AUDIT/LOG Menu Functions

The AUDIT/LOG menu provides the following functions:

#	Function	Description
(130)	**Get Config**	Shows the current configuration for audit logging.
(131)	**Set Config**	Set the audit logging configuration parameters. This command allows you to configure: > Which events are captured in the log > The log rotation interval
(132)	**Verify Logs**	This command displays details for the indicated file, or verifies the audit log records in the specified range from the named file.
(133)	**Get Time**	This command displays the current HSM time.
(134)	**Set Time**	This command synchronizes the HSM time to the host time. This is especially useful when the host computer is synchronized by NTP, or by local drift correction. This ensures that the log times of HSM events coincide with file creation and update events in the host file system.
(135)	**Import Secret**	This command imports an audit log secret that was previously exported.
(136)	**Export Secret**	This commands exports the audit logging secret to the user's local directory for import to another HSM.
(137)	**Init Audit**	This command initializes the Audit role on the HSM. An audit domain and role password (for password-authenticated HSMs) or white Audit PED key (for PED-authenticated HSMs) are attached. This command destroys any previously existing Audit role on the HSM.
(138)	**Get Status**	This command displays the audit logging information for the indicated HSM.
(139)	**Log External**	This Luna extension to PKCS#11 allows a user application to insert text into the log record stream. This command logs a string of the user's choice to the audit log file.

CA Menu Functions

The CA menu provides the following functions:

#	Function	Description
(70)	**Set Domain**	(Not for Luna HSM 7)
(71)	**Clone Key**	(Not for Luna HSM 7)
(72)	**Set MofN**	(Not for Luna HSM 7)

#	Function	Description
(73)	**Generate MofN**	(Not for Luna HSM 7) This option allows you to generate MofN authentication splits, or secret shares. You can generate up to 16 shares (N), and you can specify how many of these shares are needed (M) in order to activate the token (up to 16).
(74)	**Activate MofN**	(Not for Luna HSM 7) This option allows you to authenticate yourself to the token using MofN secret shares generated by option **(73) Generate MofN**. You must activate MofN on a token on which MofN has been generated, or you are unable to perform any cryptographic operations with the token.
(75)	**Generate Token Keys**	(Not for Luna HSM 7) Some tokens have the ability to support customer loaded certificates used for key cloning. If your token supports this feature, and you wish to use your own key cloning certificates (rather than the default certificates provided by Thales), the first step is to Generate token keys. **NOTE** If you do this, you are not able to clone to any other Luna CA tokens except those containing your own certificate.
(76)	**Get Token Cert**	(Not for Luna HSM 7) This option is the next step in loading your own key cloning certificate onto the token. This action is done after option **(75) Generate Token Keys**.
(77)	**Sign Token Cert**	(Not for Luna HSM 7) This option is the final step to load a customer key cloning certificate to the token. This step is done after options **(75) Generate Token Keys** and **(76) Get Token Cert**.
(78)	**Generate CertCo Cert**	(Not for Luna HSM 7) Generate a special-purpose certificate for CertCo application.
(79)	**Modify MofN**	(Not for Luna HSM 7) Modifies the secret splitting vector on a token.
(86)	**Duplicate MofN Keys**	(Not for Luna HSM 7) Create duplicates (copies) of all MofN secret splits.
(87)	**Deactivate MofN**	Decache the MofN data.
(88)	**Get Token Certificates**	Extract one of the following certificates from the HSM. You must supply the type and filename of the certificate you want to extract: > Root certificate > Hardware origin certificate > ECC hardware origin certificate > TWC (token wrapping certificate) version 1, 2, or 3. > CITS device authentication certificate

#	Function	Description
(112)	**Set Legacy Cloning Domain**	This option sets the legacy Cloning Domain, from a legacy token, into association with the modern cloning domain attached to a current-model Luna HSM, to allow migration of token objects from legacy HSMs.

CLUSTER EXECUTION Menu Functions

The CLUSTER EXECUTION menu and its functions are to be deprecated in a future release, and are not usable.

HIGH AVAILABILITY RECOVERY Menu Functions

The HIGH AVAILABILITY RECOVERY menu provides the following functions:

#	Function	Description
(50)	**HA Init**	This option is used for HA Login setup and requires that an RSA key pair has been previously created on the primary partition, the private key has been cloned to the user space (and optionally to the SO spaces) of all tokens within that environment. This option requires the handle to the session (of the user that owns the key pair), and the handle to the login private key itself. > If you are using Luna HSM Client 10.2.0 or older, this option prompts you to specify the HA Login private key handle. > If you are using Luna HSM Client 10.3.0 or newer and • the target HSM is running Luna HSM Firmware 7.4.2 or older, this option prompts you for the HA Login public or private key • the target HSM is running Luna HSM Firmware 7.7.0 or newer, this option prompts for initialization or revocation of the HA Login credentials. If you are initializing, then it will prompt for the HA Login private key PKC chain or the HA Login private key handle. If the HA Login private key already exists on the partition, the PKC chain will be pulled directly from the HA Login private key. If the HA Login private key does not already exist on the partition, the PKC chain can be obtained by using **ckdemo** to display all of the HA Login private key object attributes. For more information, refer to "OBJECT MANAGEMENT Menu Functions" on page 52.
(51)	**HA Login**	This option initiates several functions, including creation of a TWC (Token Wrapping Certificate) blob and HA Login Challenge (secondary token in the current HA domain) and Acceptance (primary token), as described in the document Extensions to PKCS#11, Cryptographic Token Interface Standard.
(52)	**HA Status**	Display the current status for a specified HA slot.

KEY Menu Functions

The KEY menu provides the following functions:

#	Function	Description
(60)	**Wrap Key**	This option allows you to encrypt a key. You must provide the encryption mechanism type, the handle of the wrapping key (used to encrypt the key), and the handle of the key to be wrapped (the one that is going to be encrypted). Wrapping of private asymmetric keys requires that the Partition Policy 0: “Allow private key wrapping” is turned on, and Policy 1: Allow private key cloning must be off. This is a partition-level action since Luna HSM Firmware 7.1.0.
(61)	**Unwrap Key**	This option allows you to import a wrapped (encrypted) key into the token. You are asked for the mechanism to be used for the unwrapping operation as well as what type of key is being unwrapped. Depending on the type of key being unwrapped, you are asked for some information about the key. Then you must provide a key handle of the token key to be used in the unwrapping (decryption) operation, and finally, give the name of the file containing the wrapped key. If the unwrapping key has an associated CKA_UNWRAP_TEMPLATE attribute, this affects the results of the operation. Note that if you are generating a key in ckdemo, the option to attach an unwrap template is disabled by default. You can enable this option in the OTHERS menu (see "OTHERS Menu Functions" on page 54).
(62)	**Generate Random Number**	This option generates a specified amount of random data. You are asked how many bytes of random data to generate, then are presented with the random value.
(63)	**Derive Key**	This option allows you to use a key derivation mechanism to derive a key on the token. There are several key derivation mechanisms to choose from, and you are presented with a menu of the choices. Depending on the key derivation mechanism, you are asked for some information about the key. If the base key used for generation includes a CKA_DERIVE_TEMPLATE attribute, the information you provide is added with the attributes in the derive template. If your information contradicts the attributes in the derive template, the derive operation fails. Note that if you are generating a key in **ckdemo**, the option to attach a derive template is disabled by default. You can enable this option in the OTHERS menu (see "OTHERS Menu Functions" on page 54).
(64)	**PBE Key Generation**	This option allows you to perform a "Password Based Encryption" key generation. This option is useful because it allows you to put the same key on multiple tokens without ever knowing the key value itself.
(65)	**Create Known Keys**	This option attempts to load a known key onto the token. However, due to policy setting on most tokens, this option is not allowed. As an alternative, it is possible to encrypt a known key and then unwrap it onto the token. See the Unwrap Key sample code provided with the SDK distribution.
(66)	**Seed RNG**	Provide a seed value to the HSM's Random Number Generator.

#	Function	Description
(67)	**EC User Defined Curves**	Set the desired attributes and point to a file containing Elliptical Curve parameters for generating EC keys.
(69)	**Translate Key**	This option allows you to re-encrypt an encrypted using a different key and/or mechanism. You are asked for the mechanism and a key handle of the token key to be used in the unwrapping (decryption) operation and the mechanism and key handle of the token key to be used in the wrapping (encryption) operation. Finally, you give the name of the file containing the wrapped key and a file to contain the newly wrapped key.

KEY AUTHORIZATION Menu Functions

The Key Authorization menu provides the following functions:

#	Function	Description
(210)	**Authorize Key**	Authorize the current key for use. Provide the previously set authentication to unlock the key for the current session.
(211)	**Set Authorization Data**	Owner of the key sets the authorization (password) for that key. This session activity is not viewable by CO, CU, LCU, or anyone other than the actual key owner. This is effectively a "change password" operation, and requires knowing the current authorization.
(212)	**Reset Authorization Data**	CO resets the authorization for the key (password) on behalf of a key owner who has lost/forgotten the authorization for her/his key. This operation does not require possession of the key authorization secret or password.
(213)	**Assign Key**	When LCO has generated a key with assigned=0, extractable=0, sensitive=1, modifiable=0, the CO can change it to state assigned=1 giving sole control over that key to the User who owns it. This allows keys to be imported to the HSM and then signed (note: once assigned, a key cannot be exported).

OBJECT MANAGEMENT Menu Functions

The OBJECT MANAGEMENT menu provides the following functions:

#	Function	Description
(20)	**Create Object**	This option allows you to create objects on the token. You can use this option to create data or certificate objects on the token. You are presented with a default template for your new object that you can change or choose to accept as default. **NOTE** Key generation is not done with this option. Use function **(45) Generate Key**.
(21)	**Copy Object**	This option allows you to make a copy of a token object and allows you to add/remove/change attributes of the object as you copy it.
(22)	**Destroy Object**	This option allows you to permanently delete a token object from the token.
(23)	**Object Size**	This option asks you for an object handle and returns the total size of the object (how much memory it is occupying on the token).
(24)	**Get Attribute**	This option asks you for an object handle and returns the attributes of that object.
(25)	**Set Attribute**	This option allows you to change the value of an attribute on an object that already exists on the token.
(26)	**Find Objects**	This option searches the token for objects that are available to you as the User or the SO (depending on which identity you used to log in). You specify a type (such as Data Objects, various Key objects, Certificate Objects, etc.). Option 6 shows all the objects on the token.
(27)	**Display Object**	This option shows all the attributes and associated values for an object on the token (if that object is available to you). **NOTE** If a key is sensitive, it contains an attribute called CKA_VALUE but this attribute is not displayed because the token does not allow this information to be exported.
(30)	**Modify Usage Count**	This option allows you to increment the current value, or specify a new value, for an object's usage counter. You are prompted for the object handle and whether you want to increment or reset the usage counter for the specified object.
(31)	**Destroy Multiple Objects**	This option allows you to permanently delete multiple token objects from the selected token.
(32)	**Extract Public Key**	This option allows you to specify a public key to extract from the HSM. The key is saved as **publickey.bin** in the current directory, overwriting any existing **publickey.bin** file. **NOTE** The Extractable attribute must be set to 1 (On) in order for a public key to be extracted from the HSM.

OFFBOARD KEY STORAGE Menu Functions

The OFFBOARD KEY STORAGE menu provides the following functions:

#	Function	Description
(101)	**Extract Masked Object**	Extracts a key off the Luna Network HSM in a masked format, into a file **masked.key**. You can rename the resulting file if you are testing with multiple extractions.
(102)	**Insert Masked Object**	Inserts an extracted, masked blob (file) back onto the Luna Network HSM. You are prompted for the name of the file, which must have been extracted from a Luna Network HSM using the same masking key (i.e., the same Luna Network HSM or a clone of it).
(103)	**Multisign With Value**	Performs the multisign function, after prompting you for the mechanism to use, the number of datablobs to be signed (limited to 5 for this demonstration command), and the data or filenames to be signed.
(104)	**Clone Object**	(Reserved for Thales use) Copies an object from the Luna Network HSM to another HSM.
(105)	**SIMExtract**	This function takes a list of object handles, extracts them using the given authorization data for protection, and returns the extracted set of objects as a single data blob. The objects can be left on the partition or destroyed, depending on the value of the delete-after-extract flag.
(106)	**SIMInsert**	This function inserts the objects contained in a previously extracted blob into the HSM, and returns the list of handles assigned to the objects.
(107)	**SimMultiSign**	This function uses the key material in a previously extracted key blob to sign pieces of data in the input data table, returning the signatures through the signature table. Note the following restrictions: > On a Luna Network HSM with Luna HSM Firmware 7.4.2 or older, the key blob must contain a single key, otherwise an error is returned. > On a Luna Network HSM with Luna HSM Firmware 7.7.0 or newer, the indicated blob must contain no more than one key/key pair that is suitable for the requested signature mechanism, otherwise an error is returned.
(118)	**Extract Object**	Extracts a key off the Luna Network HSM into a file.
(119)	**Insert Object**	Inserts an extracted blob (file) back onto the Luna Network HSM.

OTHERS Menu Functions

The OTHERS menu provides the following functions:

#	Function	Description
(90)	**Self Test**	Calls `CA_PerformSelfTest()` to perform any of 3 selected tests
(92)	**Get AppID**	Displays the AppID associated with the current process. To test, see 97 - Set App ID
(93)	**Utilization Metrics**	[1] Show Utilization Metrics (Requires a session already open against an application partition). [2] Reset Utilization Metrics. [0] Exit
(94)	**Open Access**	Creates a token access ID that is independent of any sessions so that the login state can be maintained even when your application exits. Used to allow the same application to return repeatedly for access without requiring a separate login each time. Remains active until closed with function **(95) Close Access** or until the token is removed.
(95)	**Close Access**	Kills the ID generated by function **(94) Open Access**.
(97)	**Set App ID**	You are prompted to type in an explicit application ID (in two parts, Major and Minor), rather than having it generated by Chrystoki. Doing so effectively causes all processes (using that Major/Minor application ID) on the machine to be recognized as the same application. Refer to the PKCS#11 Extensions document.
(98)	**Options**	This item allows you to change some default options of the **ckdemo** program. You can turn off help (which prevents the entire menu from being displayed after each command), or select the type of session you wish **(1) Open Session** command to use. Use Option 0 to exit this menu and return to the **ckdemo** main menu. For a list of these options, see "(90) Options" below.
(100)	**LKM Commands**	This option is deprecated and is to be removed from a future release; it is not usable.

(90) Options

Tests 1, 2, and 3 are available; the rest are reserved.

Option	Description
1 - H/W Test	Perform HSM hardware test
2 - Crypto Test	Perform test of cryptographic operations
3 - RNG Test	Perform test of Random Number Generator

Option	Description
4 - Perf mode on (us)	Internal use only; requires the HSM to have special test firmware.
5 - Perf mode on (ns)	Internal use only; requires the HSM to have special test firmware.
6 - Perf mode off	Internal use only; requires the HSM to have special test firmware.
7 - Cryptographic Algorithm Self Tests	Internal use only; requires the HSM to have special test firmware.
8 - Sentry off	Internal use only; requires the HSM to have special test firmware.
9 - Sentry on	Internal use only; requires the HSM to have special test firmware.
10 - Inject error: exit()	Internal use only; requires the HSM to have special test firmware.
11 - Inject error: raise()	Internal use only; requires the HSM to have special test firmware.
12 - Inject error: kernel oops	Internal use only; requires the HSM to have special test firmware.
13 - Inject error: infinite loop	Internal use only; requires the HSM to have special test firmware.
14 - List all enabled Sentry PKA engines (0:5)	Internal use only; requires the HSM to have special test firmware.
15 - Disable a Sentry PKA engine (0:5)	Internal use only; requires the HSM to have special test firmware.
16 - Enable a Sentry PKA engine (0:5)	Internal use only; requires the HSM to have special test firmware.
0 - To cancel	Return to **ckdemo** main menu

(98) Options

Use option 16 if HSM firmware is newer than version 6.22.0 and you wish to use CKR_TEMPLATE_INCONSISTENT.

Option	Description (Default)	Alternate
1 - Open Session Type	Always R/W and Serial	User selectable
2 - Display Help	Always	On demand
3 - PIN path	User supplied ASCII password	Selectable
4 - Echo input	Disabled	On all commands and data
5 - Sleep for n seconds	Sleep for n seconds after writing special instructions to stderr	Enter a number of seconds to sleep. Then enter the desired instructions. Finish entering instructions with a period (.) alone on a line
6 - KCV Default	User supplied KCV Domain	Selectable
7 - MofN path	User supplied MofN path	Selectable
8 - Show Response Code	SHOW_RESPONSE_BEFORE_MENU	SHOW_RESPONSE_ BEFORE_AND_AFTER_ MENU
9 - Input data for sign/derive	Input from keyboard	Input from file
10 - Object Usage Counters	Disabled	Selectable
11 - GCM IV Source	External	Internal
12 - ECIES Parameters	Use default (XOR with HMAC_SHA1)	Selectable
13 - X9.31 Signatures	Allow X9.31 generated keys only	Allow non-X9.31 generated keys
14 - Multipart enc/dec/sig/ver	Use single part operations	Use multi-part operations
15 - Use Old Enc/Dec Menu	Use old Encrypt/Decrypt menu	Use new Encrypt/Decrypt menu
16 - Role Support	Enhanced roles - use with roles as they are implemented with PSO-capable firmware (f/w 6.22.0 and newer)	Use HSM with legacy Luna roles (as found with f/w previous to v6.22.0)
17 - OAEP Hash Params	Use default (SHA1 Digest and MGF1)	Selectable

Option	Description (Default)	Alternate
18 - Array Template Attributes	Do not use array template attributes	Use array template attributes
0 - Finished	Return to **ckdemo** main menu	

PED INFO Menu Functions

The PED INFO menu provides the following functions:

#	Function	Description
(120)	**Set PED Info**	Specify the Luna PED (local or remote) that is associated with the HSM in a specific slot.
(121)	**Get PED Info**	Display information describing the Luna PED that is associated with the HSM in a specific slot.
(122)	**Init RPV**	Create a Remote PED Vector, and imprint it onto an orange Remote PED Key (RPK), to allow Luna PED functions with a remotely located Luna HSM (which must also have the same RPV).
(123)	**Delete RPV**	Remove the Remote PED Vector from the current HSM. Disallows Remote PED operation for this HSM until (if) a new RPV is created or an existing RPV is acquired from an imprinted RPK.

POLICY Menu Functions

The Policy menu provides the following functions:

#	Function	Description
(53)	**Show Partition Policies**	This command displays the partition-level capability and policy settings for the partition and User. The list includes all available policies, including those that an end user cannot modify.
(54)	**Set Partition Policies**	This command sets a user policy on the partition.

#	Function	Description
(55)	**Show HSM Policies**	This command displays the HSM-level capability and policy settings for the HSM. The list includes all available policies, including those that an end user cannot modify. **NOTE** The output of this command differs considerably, depending on the firmware version of the HSM in the current slot.
(56)	**Set HSM Policies**	This command sets HSM-level policies that are non-destructive.
(57)	**Set Destructive HSM Policies**	This command sets HSM-level policies that are destructive. That is, setting these policies forces the HSM to be wiped (reinitialized), destroying all contents.

SCRIPT EXECUTION Menu Functions

The SCRIPT EXECUTION menu and its functions are deprecated and will be removed in a future release, and are not usable.

SECURITY Menu Functions

The SECURITY menu in CKDemo provides the following functions:

#	Function	Description
(40)	**Encrypt File**	This option allows you to encrypt a file. You are asked which encryption mechanism you wish to use, then the path and filename of the file to be encrypted, and finally the key handle of the key to be used in the encryption operation.
(41)	**Decrypt File**	This option allows you to decrypt an encrypted file. You are asked for the encryption mechanism to use to decrypt the file, path and name of the file to be decrypted, and the handle of the key to be used for the decryption.
(42)	**Sign**	This option signs a string of data using a token signing mechanism. You are prompted for the signing mechanism that you wish to use, the data to be signed, and the key handle of the signing key (private key when using a Private/Public key pair). **NOTE** This option takes in a string of data to be signed from the keyboard, rather than a filename of a file containing the data (like encryption does). The signature is saved to a file called **SIGN.BIN**.
(43)	**Verify**	This option verifies a signature against a string of data. You are prompted for the mechanism to be used for verification, the data to be verified and the key handle of the verification key. The signature is read from the file **SIGN.BIN** that is generated during the sign operation.

#	Function	Description
(44)	**Hash**	File This option prompts for the hashing mechanism to be used, and the name of the file to be hashed. The hash value is saved to a file called **DIGEST.HSH** at the end of the operation.
(45)	**Simple Generate Key**	This option performs key generation on the token. You are presented with a menu of possible key types. Depending on the key type being generated, you are asked a list of question about the attributes of the key(s). If the option to use array attributes is enabled through the OTHERS menu, you are presented with the option to use and edit a CKA_UNWRAP_TEMPLATE or CKA_DERIVE_TEMPLATE. These templates affect the **(61) Unwrap Key** and **(63) Derive Key** functions.
(46)	**Digest Key**	This option prompts for a digest mechanism and a key handle. The key value is digested using the selected mechanism.

SRK Menu Functions

> **NOTE** These functions are not applicable to Luna HSMs with Luna HSM Firmware 7.0.1 or newer.

The SRK menu provides the following functions:

#	Function	Description
(200)	**SRK Get State**	Shows the current state of the Master Tamper Key.
(201)	**SRK Restore**	Gets the external split (SRK) of the Secure Recovery Vector from a connected Luna PED, combines it with the internally-stored split, to regenerate the SRV, and re-validates the MTK
(202)	**SRK Resplit**	Performs a new split of the Secure Recovery Vector and places the external portion of the split onto a purple PED key (called the Secure Recovery Key or SRK).
(203)	**SRK Zeroize**	Zeroize the SRK. This action simulates a hardware tamper.
(204)	**SRK Enable/ Disable**	Enable splitting of the Secure Recovery Vector into an internal (to the HSM) portion and an external portion (stored on a purple PED key). Or, disables that function by bringing the external split back into the HSM (requires Luna PED and the purple PED key with the correct SRV split on it - that purple key then becomes invalid).

TOKEN Menu Functions

The TOKEN menu provides the following functions:

#	Function	Description
(1)	**Open Session**	Before you can manipulate objects or perform cryptographic operations on a token, you must have an open session on that token. This command prompts you for the number of the slot on which to open the new session. By default, an exclusive, Read/Write session is opened. If you would like to open a read only or non-exclusive session, you must use the **(98) Options** function and specify that you want to be prompted for session types.
(2)	**Close Session**	Once you are finished using a session, the session should be closed. The **(2) Close Session** function allows you to close a single session, or to close all the sessions on a specific token.
(3)	**Login**	Once a session is opened, you usually log on to the token. You have a choice between logging on as: > Partition SO (PO) - initialize other roles and do partition administration operations, unblock blocked PKA keys > Crypto Officer (CO) - created by SO, can perform crypto operations including creating/deleting/ backing up keys > Limited Crypto Officer (LCO) - created by CO, can generate/delete keys, SIMExtract/SIMInsert, derive and wrap/unwrap (part of Per Key Authorization), cannot unblock > Crypto User (CU) - created by CO, read-only crypto operations
(4)	**Logout**	When you are finished with the token, you should first log out, then close the session.
(5)	**Change PIN**	(Not for Luna Network HSM) This option lets you change the logon password (the PIN) of the currently logged in user. You must supply both the old PIN and the new PIN to complete the operation.
(6)	**Init Token**	(Not for Luna Network HSM) This option allows you to reset a token to its initial state. You are prompted for the following: > The slot containing the token to be initialized > The token label (which is simply a text string that you can use for Token Identification) > A new password for the Partition SO Token initialization performs the following actions: > Wipes out any token objects (Keys, certificates, etc) > Clears the user PIN (so that it must be reset by the Partition SO) > Sets the SO PIN to the value that you have specified
(7)	**Init PIN**	(Not for Luna Network HSM) This command is used to create a user (and thus overwrites an existing user) and is run when you are logged in as the Partition SO.

#	Function	Description
(8)	**Mechanism List**	This option gives a list of all the encryption/authentication/hashing/key-generation mechanisms supported by the token. If you want to know if the token supports a specific type of encryption, you can check for it in the mechanism list.
(9)	**Mechanism Info**	This option allows you to query a specific mechanism to find such information as supported key sizes. You are asked for the Mechanism type, which is a numeric value representing the mechanism (these numeric values are given when you request a mechanism list).
(10)	**Get Info**	This option returns basic information on the Dynamic Library that is being used to talk to the token. None of this information is token specific, and it can be viewed even if there is no token present.
(11)	**Slot Info**	This option gives specific information on a card slot. The slot description and slot ID are given, as well as some flags to represent if a token is present.
(12)	**Token Info**	This option gives information on a token in a specific slot, including the following: > Token Label > Token Manufacturer > Token Model > Token Flags > Session Count > Min and Max PIN Lengths > Private memory size/free > Public memory size/free
(13)	**Session Info**	This option gives information on an open session. You must have at least one session opened to query session information. For a particular session you can find the session handle, the slot ID, the session state, and any associated session flags.
(14)	**Get Slot List**	This option returns a list of card slots available on the system. You are given the option to view all slots, or just the slots which contain tokens.
(15)	**Wait for Slot Event**	Runs CK_WaitforSlotEvent (from PKCS#11 Extensions).
(18)	**Factory Reset**	This option resets the HSM to its factory settings.
(19)	**Clone MofN**	(Not for Luna Network HSM) Copy a clonable secret-splitting vector from one token to another.
(33)	**Token Insert**	(For Luna USB HSM 7) This option signals the HSM or local workstation that a token will be inserted. Insert the token to begin performing operations with it.
(34)	**Token Delete**	(For Luna USB HSM 7) This option deletes the token in a specific slot.

#	Function	Description
(36)	**Show Roles**	This option lists the roles currently configured on the token in a specific slot.
(37)	**Show Role Configuration Policies**	This option lists the role configuration policies currently in effect for the named role on the token in a specific slot.
(38)	**Show Role State**	This option shows the state of the named role. Information given includes: > Primary authentication type > Secondary authentication type > Failed login attempts before lockout > Failed change password attempts before lockout (only shown when using Luna Network HSM Appliance Software 7.7.0 and newer, Luna HSM Firmware 7.7.0 and newer, and Luna HSM Client 10.3.0 and newer) > Init status
(39)	**Get OUID**	This option retrieves the OUID (Object Unique Identifier) of a token in a specific slot.
(58)	**HSM Zeroize**	This option zeroizes the HSM, removing all partitions and keys. HSM zeroization does not destroy the RPV or Auditor role.
(59)	**Token Zeroize**	This option zeroizes the token in a specific slot, removing all keys and objects.

CHAPTER 3: multitoken

multitoken is a simple demonstration tool that allows you to perform basic cryptographic functions on a Luna HSM. It allows you to specify an operation, and one or more “slots” or HSM Partitions on which to perform that operation. The **multitoken** utility runs the operations and returns a summary of the results.

> **NOTE** This is a general-purpose tool intended for use across Luna HSM versions. It might reference mechanisms and features that are not available on all Luna products.

Accessing multitoken

The **multitoken** utility is accessed via the command line.

To access the multitoken utility

1. Open a console window.
2. Go to the Luna HSM Client installation folder/directory:

Windows	C:\Program Files\SafeNet\LunaClient
Linux/Unix	/usr/safenet/lunaclient/bin
Client downloaded from Data Protection on Demand	<client_unzip_location>

3. Launch the **multitoken** utility:

   ./multitoken

Syntax

multitoken-mode <mode> {**-slots** <slot_list> | **-nslots** <slot_threads>} [options...]

Argument(s)	Shortcut	Description
-alarm <secs>	**-al**	Sound periodic alarm (every <secs> seconds) if error occurs.
-applytochild	**-atc**	Apply the PerKeyAuth settings to any child keys that are created (ie. derived or unwrapped as part of the test). [for PKA]
-assigned	**-as**	Generate keys as assigned (CKA_ASSIGNED=1). [for PKA]
-blob <blob_count>	**-b**	Number of data blobs to be signed during each multisign operation.
-curve <curve_num>	**-crv**	ID number of ECC curve. If user-defined (99), then must specify **-parmfile**.

Argument(s)	Shortcut	Description
-delayop <secs>	**-do**	Delay the operation performed by each thread by the specified number of seconds. Value must be larger than "0".
-destroyafter	**-da**	Destroys created objects on the HSM only after test completes.
-destroyafterbulk	**-dab**	Destroys created objects on the HSM only after test completes using DestroyMultipleObjects.
-eciesdata <filename>	**-ecd**	**ECIES SHIM mode:** Specifies the file that contains the plaintext data to use. **Non-SHIM ECIES mode:** Specifies the file to receive the plaintext data used. See "Notes" on page 78 for details on using SHIM and non-SHIM ECIES modes.
-eciesenc <filename>	**-ece**	**ECIES SHIM mode:** Specifies the file that contains the encrypted data. **Non-SHIM ECIES mode:** Specifies the file to receive the encrypted data. See "Notes" on page 78 for details on using SHIM and non-SHIM ECIES modes.
-ecieskey <filename>	**-eck**	**ECIES SHIM mode:** Specifies the file that contains the DER-encoded private key. **Non-SHIM ECIES mode:** Specifies the file to receive the DER-encoded private key. See "Notes" on page 78 for details on using SHIM and non-SHIM ECIES modes.
-eciesscheme	**-esch**	Encryption scheme to use for ECIES AES modes: 0 = AES_CBC_PAD (default), 1 = AES_CTR.
-enddate <YYYYMMDD>	**-end**	Validity end date for key, in YYYYMMDD format.
-force	**-f**	Avoid prompts for responses.
-gcmaad <bytes>	**-gad**	Specify the length of the AAD data used for GCM/GMAC. The AAD data can not be larger than 1024 bytes.
-gcmiv <bits>	**-giv**	Specify the length of the IV (in bits) to be used for GCM/GMAC. **Valid values:** 0,96,128
-haaidhigh <value>	**-hah**	Specify the AppID High value.
-haaidlow <value>	**-hal**	Specify the AppID Low value.

Argument(s)	Shortcut	Description
-haaidoffset <value>	**-hao**	Specify the AppID Low value increment for each thread.
-haslot <value>	**-has**	Specify the slot id for the secondary.
-halogout <value>	**-hat**	Specify that the session should be logged out.
-haclosesess	**-hac**	Specify that the session should be closed.
-hacloseappid <value>	**-haa**	Specify the AppID that should be closed.
-help	**-h**	Display help information only.
-kdfchoice <kdf_index>	**-kdf**	Select key derivation function - specify choice list index.
-kdfscnt <counter_index>	**-kds**	Select key derivation session counter type - specify choice list index.
-kekreplace	**-kre**	Specifies that a KEK replacement should be requested after the specified number of iterations. This option only works with DES3/AES ECB/CBC tests. It will be ignored for all other test modes.
-key <key_size>	**-k**	Size of key: asymmetric in bits (default = 1024 for RSA, 2048 for DSA). Symmetric in bytes (i.e. 16, 24, 32 for AES/ARIA).
-keyauthtype <key_index>	**-kat**	Specify the type Per Key Authorization test to performed. [for PKA] 1 = Authorize the key once and use it many times 2 = Authorize the key once, use it once, rescind the key
-keyauthdata <data>	**-kad**	Specify the authorization data to use for the key. [for PKA]
-keychoice <key_index>	**-kc**	Select key type to derive/generate - specify choice list index.
-keyderiv <keysize>	**-kde**	Size of key to derive with (ex. 1024 for X9.42 Diffie Hellman).
-kwicv	**-kiv**	Use external ICV for the key wrap mechanism.
-limitedco <keysize>	**-lco**	Login as the Limited Crypto Officer (default is Crypto Officer).
-logfile <filename>	**-l**	File for results logging.
-mode <mode>	**-m**	Operating mode. See mode values available below.
-multipartsig	**-msig**	Use multipart signatures.

Argument(s)	Shortcut	Description
-nodec	**-nod**	Decryption operation will not be performed. Only symmetric and asymmetric encryption will be performed and measured.
-nodestroy	**-n**	Leaves created objects on the HSM after test completes.
-noenc	**-noe**	Perform only one encryption operation. Only symmetric and asymmetric decryption will be performed and measured.
-nosign	**-nos**	Perform only one sign operation. Only verify will be performed and measured.
-nounwrap	**-nou**	Unwrapping operation will not be performed. Only wrapping will be performed and measured.
-noverify	**-nov**	Verify operation will not be performed. Only sign will be performed and measured.
-noverifyr	**-nvr**	Do not verify decryption results.
-nowrap	**-now**	Perform only one wrapping operation. Only unwrapping will be performed and measured.
-nslots <slot_threads>	**-ns**	Create multiple threads on the same slot(s). Specify <slot>x<number of threads>, with multiple slots separated by commas. The example below creates 5 threads on slot 1 and 20 threads on slot 2: Example: **-nslots 1x5,2x20** You must specify either this option or **-slots**. See "-slots <slots>" on the next page.
-objcount <objnum>	**-obj**	Interpretation of this parameter depends on test mode. If this is a find objects test, it specifies the number of objects to create. If this parameter is unspecified, a default of **1000** is used. For symgen operations, this specifies the total number of objects to create. If this parameter is unspecified, a default of **0** is used, which means 'unlimited'. For rsakeygen operations, this specifies the total number of key pairs to create. Again, if this parameter is unspecified, a default of **0** is used, which means 'unlimited'.
-objtype	**-objt**	Type of object to create for object creation/deletion test (mode objectcreation). 0 = data object (default), OR public key: 1=DSA, 2=RSA, 3=EC Montgomery, 4=EC Edwards, 5=ECDSA
-outputlen <length-in-bytes>	**-ol**	Size of output length in bytes for SHAKE.

Argument(s)	Shortcut	Description
-packet <packet_size>	**-p**	Size of packet used in operation.
-parmfile <param_file>	**-prm**	File for EC curve parameters or OAEP source data (0 = none for OAEP).
-password <password>	**-pwd**	Specify password to use for token.
-pbkd2prf		Specify the type of PRF to use for PBKD2-based key derivation.
-ped <0/1>	**-ped**	Specifies the type of Luna PED connection. This applies only to the first HSM slot to be specified using the **-slots** option. **Valid values:0**(local),**1**(remote)
-prftype <type>	**-prf**	Specify the type of PRF to use for PRF based key derivation.
-scroll	**-scr**	Scroll the output instead of overwriting it each time.
-session	**-ses**	Use session objects instead of token objects.
-sharefile <filename>	**-shf**	Shared data file used for operation.
-silent	**-sil**	Disables system "beep" that is generated when a error occurs.
-simblobuse	**-sbu**	Specify operation to use keys from SIM blobs.
-slots <slots>	**-s**	List of slots to use (slot numbers separated by commas). List the same slot multiple times to create multiple threads on that slot. The example below creates 2 threads on slot 1 and 3 threads on slot 2: Example: **-slots 1,1,2,2,2** To create many threads on the same slot, use **-nslots** instead. See "-nslots <slot_threads>" on the previous page.
-startdate <YYYYMMDD>	**-sta**	Validity start date for key in format <YYYYMMDD>.
-subprime <size>	**-sub**	Size of the subprime in bits.
-sym_c_u_d <value>	**-scud**	For each test loop, create the key(s), use the key(s) and then delete the key(s). Only supported for symmetric enc/dec/sig/verifywhen **-nosign**, **-noverify**, **-noenc** and **-nodec** are NOT used. This argument takes a value that indicates how many times the key(s) should be used before it is deleted and a new key(s) is created. This value must be 1 or larger.

Argument(s)	Shortcut	Description
-symm <mechanism>	**-sym**	Select symmetric key mechanism for symderive/pbegen or key choice for symgen (can also use **-keychoice**).
-template	**-tp**	Attaches a generic unwrap template or derive template for the wrapunwrap or symderive mode respectively.
-timed <secs>	**-t**	Fixed amount of time to run (seconds).
-usage <uses>	**-u**	Number of times a key is allowed to be used.
-verbose	**-v**	Show all thread performances. Default is only first and last threads.
-wkeysize	**-wkz**	Size of RSA private key to be wrapped for aeskwp wrap/unwrap test.
-wkeytype	**-wk**	Type of key to be wrapped for key wrap/unwrap test using aeskwp. (mode aeswrapkwp/aeswrapgcm). 0 = DES3 (default), 1 = RSA private key

Operating Modes

The following table lists the available operating modes for the **multitoken** utility. The operating mode is specified using the **-mode** parameter.

Mode	Description
aescmac	AES CMAC sign
aesenc	AES ECB encrypt
aesenccbc	AES CBC encrypt
aesenccfb8	AES CFB8 encrypt
aesenccfb128	AES CFB128 encrypt
aesencctr	AES CTR encrypt
aesencgcm	AES GCM encrypt
aesenckw	AES KW encrypt
aesenckwp	AES KWP encrypt

Mode	Description
aesencofb	AES OFB encrypt
aesgmac	AES GMAC sign
aesmac	AES MAC sign
aeswrapkw	AES KW wrap
aeswrapkwp	AES KWP wrap
aesxts	AES XTS encrypt
ariacmac	ARIA CMAC sign
ariaenc	ARIA ECB encrypt
ariaenccbc	ARIA CBC encrypt
ariaenccfb8	ARIA CFB8 encrypt
ariaenccfb128	ARIA CFB128 encrypt
ariaencctr	ARIA CTR encrypt
ariaencofb	ARIA OFB encrypt
ariamac	ARIA MAC sign
bip32childkeyderive	BIP32 Child Key (Normal) Derivation
bip32childkeyhardenedderive	BIP32 Child Key (Hardened) Derivation
bip32childkeypublicderive	BIP32 Child Key (public to public) Derivation
bip32gbcssha256sigver	SHA256 BIP32-GBCS sign
bip32masterkeyderive	BIP32 Master Key Derivation
bip32sha1sigver	SHA1 BIP32 sign
bip32sha224sigver	SHA224 BIP32 sign
bip32sha256sigver	SHA256 BIP32 sign
bip32sha384sigver	SHA384 BIP32 sign

Mode	Description
bip32sha512sigver	SHA512 BIP32 sign
bip32sigver	BIP32 sign
des3enccfb8	DES3 CFB8 encrypt
des3enccfb64	DES3 CFB64 encrypt
des3encctr	DES3 CTR encrypt
des3encofb	DES3 OFB encrypt
descmac	DES3 CMAC sign
desenc	DES3 ECB encrypt
desenccbc	DES3 CBC encrypt
desmac	DES3 MAC sign
desx919mac	DES3 X919 MAC sign
dhparamsgen	DH Domain Parameter Generation
dsakeygen	DSA Key Generation
dsaparamsgen	DSA Domain Parameter Generation
dsasigver	DSA bare sign
dukptderive	DUKPT key derivation
ecdhcderive	ECDH Cofactor derive key
ecdhderive	ECDH derive key
ecdhderivewrapnew	ECDH derive and wrap new
ecdhderivewrapold	ECDH derive and wrap old
ecdsagbcssha256sigver	SHA256 ECDSA-GBCS sign
ecdsakeygen	ECDSA Key Generation
ecdsakeygenwextrabits	ECDSA Key Gen with Extra Bits

Mode	Description
ecdsasha1sigver	SHA1 ECDSA sign
ecdsasha224sigver	SHA224 ECDSA sign
ecdsasha256sigver	SHA256 ECDSA sign
ecdsasha384sigver	SHA384 ECDSA sign
ecdsasha512sigver	SHA512 ECDSA sign
ecdsasha3-224sigver	SHA3-224 ECDSA sign
ecdsasha3-256sigver	SHA3-256 ECDSA sign
ecdsasha3-384sigver	SHA3-384 ECDSA sign
ecdsasha3-512sigver	SHA3-512 ECDSA sign
ecdsasigver	ECDSA sign
ecedwardskeygen	EC Edwards Key Generation
eciesaes128hmacsha256	ECIES AES-128 enc/dec with HMAC SHA256
eciesaes128hmacsha256shared	ECIES AES-128 enc/dec with HMAC SHA256 and shared data
eciesaes192hmacsha384	ECIES AES-192 enc/dec with HMAC SHA384
eciesaes192hmacsha384shared	ECIES AES-192 enc/dec with HMAC SHA384 and shared data
eciesaes256hmacsha512	ECIES AES-256 enc/dec with HMAC SHA512
eciesaes256hmacsha512shared	ECIES AES-256 enc/dec with HMAC SHA512 and shared data
eciesdes3hmacsha224	ECIES DES3 enc/dec with HMAC SHA224
eciesdes3hmacsha224shared	ECIES DES3 enc/dec with HMAC SHA224 and shared data
eciesshimaes128hmacsha256	ECIES AES-128 with HMAC SHA256 decrypt
eciesshimaes128hmacsha256shared	ECIES AES-128 with HMAC SHA256 and shared data decrypt
eciesshimaes192hmacsha384	ECIES AES-192 with HMAC SHA384 decrypt
eciesshimaes192hmacsha384shared	ECIES AES-192 with HMAC SHA384 and shared data decrypt

Mode	Description
eciesshimaes256hmacsha512	ECIES AES-256 with HMAC SHA512 decrypt
eciesshimaes256hmacsha512shared	ECIES AES-256 with HMAC SHA512 and shared data decrypt
eciesshimdes3hmacsha224	ECIES DES3 with HMAC SHA224 decrypt
eciesshimdes3hmacsha224shared	ECIES DES3 with HMAC SHA224 and shared data decrypt
eciesshimxorhmacsha1	ECIES XOR with HMAC SHA1 decrypt
eciesshimxorhmacsha1shared	ECIES XOR with HMAC SHA1 and shared data decrypt
eciesxorhmacsha1	ECIES XOR enc/dec with HMAC SHA1
eciesxorhmacsha1shared	ECIES XOR enc/dec with HMAC SHA1 and shared data
ecmontkeygen	EC Montgomery Key Generation
eddsanaclsha1sigver	SHA1 EDDSA NaCl sign
eddsanaclsha224sigver	SHA224 EDDSA NaCl sign
eddsanaclsha256sigver	SHA256 EDDSA NaCl sign
eddsanaclsha384sigver	SHA384 EDDSA NaCl sign
eddsanaclsha512sigver	SHA512 EDDSA NaCl sign
eddsanaclsigver	EDDSA NaCl sign
eddsasha1sigver	SHA1 EDDSA sign
eddsasha224sigver	SHA224 EDDSA sign
eddsasha256sigver	SHA256 EDDSA sign
eddsasha384sigver	SHA384 EDDSA sign
eddsasha512sigver	SHA512 EDDSA sign
eddsaphsigver	EDDSA PH sign
eddsasigver	EDDSA sign
extractinsert	Extract Insert masked objects

Mode	Description
findobject	Find objects
haLogin	HA Login
kcdsakeygen	KCDSA Key Generation
kcdsasha1sigver	SHA51 KCDSA sign
kcdsasha1sigvernopad	SHA1 KCDSA NO-PAD sign
kcdsasha224sigver	SHA224 KCDSA sign
kcdsasha224sigvernopad	SHA224 KCDSA NO-PAD sign
kcdsasha256sigver	SHA256 KCDSA sign
kcdsasha256sigvernopad	SHA256 KCDSA NO-PAD sign
kcdsasha384sigver	SHA384 KCDSA sign
kcdsasha384sigvernopad	SHA384 KCDSA NO-PAD sign
kcdsasha512sigver	SHA512 KCDSA sign
kcdsasha512sigvernopad	SHA512 KCDSA NO-PAD sign
kcdsasigver	HAS160 KCDSA 1024-bit sign
kcdsasigvernopad	HAS160 KCDSA NO-PAD 1024-bit sign
keccak-224	KECCAK-224 Hashing
keccak-256	KECCAK-256 Hashing
keccak-384	KECCAK-384 Hashing
keccak-512	KECCAK-512 Hashing
md5	MD5 Hashing
milenage	3GPP Milenage AUTN
multisignvalue	Multisign w/ masked key **NOTE:** not used; deprecated.
ntlsEcho	Test NTLS/SSL Throughput

Mode	Description
objectcreation	Create/delete object
openclosesession	Open/close session
pbegen	PBE key generation
randgen	Random number generation
rc4enc	RC4 encrypt
rsa1863auxprimekeygen	RSA FIPS 186-3 using Auxiliary Primes key generation
rsa1863primekeygen	RSA FIPS 186-3 using Primes key generation
rsaenc	RSA encrypt
rsakeygen	RSA key generation
rsaoaepenc	RSA OAEP encrypt
rsapkcsenc	RSA PKCS encrypt / decrypt
rsasigver	RSA sign
rsax931keygen	RSA X9.31 key generation
rsax931sigver	X9.31 RSA sign
seedcmac	SEED CMAC sign
seedenc	SEED ECB encrypt
seedenccbc	SEED CBC encrypt
seedencctr	SEED CTR encrypt
seedmac	SEED MAC sign
sha1	SHA-1 Hashing
sha1dsasigver	SHA1 DSA sign
sha1hmac	SHA1 HMAC sign
sha1rsapsssigver	SHA1 RSA PSS sign

Mode	Description
sha1rsasigver	SHA1 with RSA sign
sha1rsax931sigver	SHA1 X9.31 RSA sign
sha224	SHA-224 Hashing
sha224dsasigver	SHA224 DSA sign
sha224hmac	SHA224 HMAC sign
sha224rsaoaepenc	SHA224 RSA OAEP encrypt
sha224rsapsssigver	SHA224 RSA PSS sign
sha224rsasigver	SHA224 with RSA sign
sha224rsax931sigver	SHA224 X9.31 RSA sign
sha256	SHA-256 Hashing
sha256dsasigver	SHA256 DSA sign
sha256hmac	SHA256 HMAC sign
sha256rsaoaepenc	SHA256 RSA OAEP encrypt
sha256rsapsssigver	SHA256 RSA PSS sign
sha256rsasigver	SHA256 with RSA sign
sha256rsax931sigver	SHA256 X9.31 RSA sign
sha384	SHA-384 Hashing
sha384hmac	SHA384 HMAC sign
sha384rsaoaepenc	SHA384 RSA OAEP encrypt
sha384rsapsssigver	SHA384 RSA PSS sign
sha384rsasigver	SHA384 with RSA sign
sha384rsax931sigver	SHA384 X9.31 RSA sign
sha512	SHA-512 Hashing

Mode	Description
sha512hmac	SHA512 HMAC sign
sha512rsaoaepenc	SHA512 RSA OAEP encrypt
sha512rsapsssigver	SHA512 RSA PSS sign
sha512rsasigver	SHA512 with RSA sign
sha512rsax931sigver	SHA512 X9.31 RSA sign
shake-128	SHAKE-128 Hashing
shake-256	SHAKE-256 Hashing
sim3extractinsert	SIM3 Extract Insert masked objects
simextractinsert	SIMExtract Insert masked objects
simmultisign	SIMMultisign w/ masked key
sm2dsasha1sigver	SHA1 SM2DSA sign
sm2dsasha224sigver	SHA224 SM2DSA sign
sm2dsasha256sigver	SHA256 SM2DSA sign
sm2dsasha384sigver	SHA384 SM2DSA sign
sm2dsasha512sigver	SHA512 SM2DSA sign
sm2dsasigver	SM2DSA sign
sm2dsasm3sigver	SM3 SM2DSA sign
sm3	SM3 Hashing
sm3hmac	SM3 HMAC sign
sm4enc	SM4 ECB encrypt
sm4enccbc	SM4 CBC encrypt
sm4enccbcpad	SM4 CBC PAD encrypt
symderive	Symmetric key derivation

Mode	Description
symgen	Symmetric key generation
tuak	3GPP Tuak AUTN
wrapunwrap	Wrap/unwrap operations
x942dhderive	X9.42 DH Derive
x942dhhybridderive	X9.42 DH Hybrid Derive
x942dhkeygen	X9.42 DH Key Pair Generation
x942dhparamsgen	X9.42 DH Domain Parameter Generation

Notes

1. If you are performing RSA operations, you have the option of specifying a key size (512, 1024, 2048, 4096, 8192). If no key size is specified, the default key size of 2048 will be used. For example:

   ```
   multitoken -mode rsasigver -key 512 -slots 1
   ```

2. If you are performing wrapunwrap operation, it will perform the following operations:
   - Generate RSA key pair and a symmetric DES key.
   - Wrap DES key with RSA public key.
   - Unwrap wrapped key above with RSA private key.
   - Verify the unwrapped key.
3. If you are performing a Multisign operation, you have the option of specifying a key size (512, 1024, 2048, 4096, 8192). If no key size is specified, the default key size of 1024 is used. For example:

   ```
   multitoken m simmultisign -ns 6x20 -pwd userpin2 -f
   ```

4. When using simmultisign with Luna HSM Firmware 7.7.0 or newer, the indicated blob must contain no more than one key/key pair that is suitable for the requested signature mechanism, otherwise an error is returned.
5. A thread is spawned to perform tests on each slot specified. A slot can be specified multiple times, in which case multiple threads will be created for the slot.
6. For modes :
   - sha256rsasigver - SHA256 with RSA
   - sha384rsasigver - SHA384 with RSA
   - sha512rsasigver - SHA512 with RSA

   If you specify a keysize on the command line (-key, any of 1024, 2048 or 4096), a file called "1024" or "2048" or "4096" is created - that is the keysize parameter is parsed as a filename to which results are saved.
7. There are two types of ECIES modes, regular modes and SHIM modes. The non-SHIM modes perform normal ECIES encryption, decryption, or both. The ECIES SHIM modes (ECIES modes with the word "shim" in them) are designed to test the ECIES SHIM implementation. The ECIES SHIM only supports decryption

operations, so it requires all input (private key, encrypted data and plaintext) to be specified as input files using the parameters **-ecieskey**, **-eciesenc** and **-eciesenc**. The input files can be created by running the non-SHIM modes, and specifying the parameters **-ecieskey**, **-eciesdata** and **-eciesenc**, which are optional for the non-SHIM modes.

For non-SHIM ECIES modes:

- **-ecieskey** -- Specifies the file to receive the DER-encoded private key.
- **-eciesdata** -- Specifies the file to receive the plaintext data used.
- **-eciesece** -- Specifies the file to receive the encrypted data.

For ECIES SHIM modes:

- **-ecieskey** -- Specifies the file that contains the DER-encoded private key.
- **-eciesdata** -- Specifies the file that contains the plaintext data to use.
- **-eciesece** -- Specifies the file that contains the encrypted data.

Named and User-defined Curves

The Luna HSMs employ named and user-defined curves. **Multitoken** supports this option, as illustrated in the following example:

```
C:\Program Files\SafeNet\LunaClient>multitoken -mode ecdsasigver -s 1,1,1,1,1,1,1,1

Prime field curves:

        [0]secp112r1  [1]secp112r2  [2]secp128r1  [3]secp128r2
        [4]secp160k1  [5]secp160r1  [6]secp160r2  [7]secp192k1
        [8]secp224k1  [9]secp224r1  [10]secp256k1 [11]secp384r1(P-384)
        [12]secp521r1(P-521)
        [13]X9_62_prime192v1 [14]X9_62_prime192v2 [15]X9_62_prime192v3
        [16]X9_62_prime239v1 [17]X9_62_prime239v2 [18]X9_62_prime239v3
        [19]X9_62_prime256v1(P-256)

Characteristic two field curves:

        [20]sect113r1 [21]sect113r2 [22]sect131r1 [23]sect131r2
        [24]sect163k1 [25]sect163r1 [26]sect163r2 [27]sect193r1
        [28]sect193r2 [29]sect233k1 [30]sect233r1 [31]sect239k1
        [32]sect283k1 [33]sect283r1 [34]sect409k1 [35]sect409r1
        [36]sect571k1 [37]sect571r1
        [38]X9_62_c2pnb163v1 [39]X9_62_c2pnb163v2 [40]X9_62_c2pnb163v3
        [41]X9_62_c2pnb176v1 [42]X9_62_c2tnb191v1 [43]X9_62_c2tnb191v2
        [44]X9_62_c2tnb191v3 [45]X9_62_c2pnb208w1 [46]X9_62_c2tnb239v1
        [47]X9_62_c2tnb239v2 [48]X9_62_c2tnb239v3 [49]X9_62_c2pnb272w1
        [50]X9_62_c2pnb304w1 [51]X9_62_c2tnb359v1 [52]X9_62_c2pnb368w1
        [53]X9_62_c2tnb431r1
        [54]Brainpool_P160r1 [55]Brainpool_P160t1 [56]Brainpool_P192r1
        [57]Brainpool_P192t1 [58]Brainpool_P224r1 [59]Brainpool_P224t1
        [60]Brainpool_P256r1 [61]Brainpool_P256t1 [62]Brainpool_P320r1
        [63]Brainpool_P320t1 [64]Brainpool_P384r1 [65]Brainpool_P384t1
        [66]Brainpool_P512r1 [67]Brainpool_P512t1

User Defined curves:

        [68]Microsoft PlayReady P-160
```

```
Please pick a curve (0-67) or enter (99) for a user defined curve:99

Please enter the filename for the EC parameters:
```

Here, you would provide the filepath to the file specifying the Elliptical Curve parameters. The format and content of the parameter file follow industry standards, and are discussed in more detail in Named Curves and User-Defined Parameters.

SKS and Per Key Auth

PerKeyAuth and/or SKS (added for Luna HSM Firmware 7.7.0 and newer) can be incorporated into other test modes.

For example

Per Key Authorization:
The "-keyauthdata" option can be used to specify authorization data (that is CKA_AUTH_DATA) that should be applied to any key(s) created to support the test.

For example, in an ECDH test, authorization data will be applied to the ECDH keys. The "-keyauthtype" can be used to specify how the authorization data is to be used.

Two main ways to use are;

- authorize the key(s) once, and use it many times and then rescind authorization, and
- authorize the key before each use and rescind authorization after each use.

For example, in an ECDH test,

- the private key can be authorized once and used for the duration of the test,
  or
- the private key can be authorized before each ECDH operation and authorization is rescinded after each operation.

For tests that use an initial set of keys and create child keys (that is, derivation or wrap/unwrap tests), "applytochild" can be used specify that the Per Key Authorization settings should also be applied to the child key (s).

For key generation test, "-keyauthdata" can be specified as the generated keys are not used. The key authorization data will be applied to the generated key.

Per Key Authorization functionality can be used only on User Partitions and only by the Crypto-Officer role.

Scalable Key Storage (SKS)
Options used for -simblobuse

SIM insertion and keys deletion test:

1. Specify both -nodec & -noenc or both -nosign and -noverify
2. Specify any crypto mode (e.g. any encrypt/decrypt or sign/verify mode)

SIM insertion, Per Key Authorization, keys deletion test:

1. Specify both -nodec & -noenc or both -nosign and -noverify

2. Specify any crypto mode (e.g. any encrypt/decrypt or sign/verify mode)
   Specify Per Key Authorization options.

SIM insertion, Per Key Authorization, crypto (enc/dec or sign/verify), keys deletion test:

1. Specify any crypto mode (e.g. any encrypt/decrypt or sign/verify mode)
2. Specify Per Key Authorization options.

Example with LCO role

```
[me@localhost bin]# ./multitoken m simextractinsert -ns 6x20 -lco -pwd userpin2 -f
multitoken (64-bit) v8.0.0-161. Copyright (c) 2019 SafeNet. All rights reserved.


Warning: Key size not specified.  Using default key size of 1024.

Initializing library...Finished Initializing
...done.


Do you wish to continue?

Enter 'y' or 'n':

Constructing thread objects.
Logging in to tokens...
  slot 6...    Serial Number 1334093726636


Please wait, creating test threads.

Test threads created successfully. Press ENTER to terminate testing.

     SIMExtract Insert masked objects:

     Using token objects.

     Logged in as Limited Crypto Officer.

               +         xfers/sec | elapsed
 6,   0  6, 19 |   total   average  | time (secs)  |     /sec    |     /sec
------ ------ | ------- ---------- | ------------ | ----------- | -----------
  10.0   10.0 |   206.4   211.923* |          350 |         0.0 |         0.0


Waiting for threads to terminate.

[me@localhost bin]# ./multitoken m simmultisign -ns 6x20 -lco -pwd userpin2 -f
multitoken (64-bit) v8.0.0-161. Copyright (c) 2019 SafeNet. All rights reserved.


Warning: Blob count not specified.  Using default key size of 1.

Initializing library...Finished Initializing
...done.
```

```
Do you wish to continue?

Enter 'y' or 'n':

Constructing thread objects.
Logging in to tokens...
  slot 6...    Serial Number 1334093726636


Please wait, creating test threads.

Test threads created successfully. Press ENTER to terminate testing.

     SIMMultisign w/ masked key : (packet size = 16 bytes)

     Using token objects.

     Logged in as Limited Crypto Officer.

               + signatures/second | elapsed
 6,  0  6, 19 |   total   average  | time (secs)  |     /sec    |     /sec
------ ------ | ------- ---------- | ------------ | ----------- | -----------
  92.8   92.0 |  1844.9  1843.808* |          145 |         0.0 |         0.0


Waiting for threads to terminate.
[me@localhost bin]#
```

CHAPTER 4: rbs

RBS implements the Remote Backup Service to remotely backup your HSMs. RBS is run on a workstation with a Luna Backup HSM connected.

RBS requires PEDclient to be running both on the RBS computer and on the host of the Luna HSM primary (the HSM being backed-up from, or being restored-to). PEDclient enables the communication link over which RBS works.

PEDclient is also used in conjunction with PEDserver to enable Remote PED, and in the case where both the Backup HSM and the Remote PED are connected to the same administrative workstation, you might legitimately have all three of RBS, PEDserver, and PEDclient running on the one system.

Syntax

rbs [**--daemon**] [**--genkey**] [**--nopassword**] [**--config**] [**--help**]

Argument(s)	Shortcut	Description
--config	**-c**	Runs RBS to select devices to support for Remote Backup.
--daemon	**-d**	Runs RBS in daemon (background) mode (Linux/UNIX only).
--genkey	**-g**	Runs RBS to generate private key/certificate for Remote Backup.
--help	**-h**	Displays help information for the rbs command.
--nopassword	**-n**	Require no password for encoded keys.

Examples

```
[admin@myluna bin] # ./rbs --config

[admin@myluna bin]#
```

```
[admin@myluna bin] # ../rbs/bin/rbs --daemon
Enter password : ********
[admin@myluna bin]#
```

```
[admin@myluna bin] # ./rbs --genkey
Enter password : ********
Verify password: ********
[admin@myluna bin]#
```

```
[admin@myluna bin] # ./rbs --nopassword

[admin@myluna bin]#
```

CHAPTER 5: salogin

Cryptographic applications that are not specifically adapted to use an HSM Server can be run using Luna HSMs, with the aid of the **salogin** utility. This section provides the settings required for some widely-used applications.

The **salogin** client-side utility is provided to assist clients that do not include the requisite HSM login and logout capability within the client application. OpenSSL, for example, can be used with HSMs, but has no inherent ability to provide credentials to the HSM.

> **NOTE** The **salogin** utility does not work with STC-enabled slots. If you require **salogin** with your applications, you must use NTLS client links.

Using salogin

Run the utility from a shell or command prompt, or include it in scripts.

Syntax

salogin {**-o** | **-c**} [**-p** <password>] [**-s** <slot> | **-l** <label>] [**-i** <hi**:**lo>] [**-u**] [**-r** <server_IP>] [**-q** <port>] [**-v**] [**-h**]

Argument(s)	Description
-c	Close application access.
-h	Display this help.
-i <hi**:**lo>	Specifies the application ID high and low components.
-l <label>	Specifies the partition label. Include either **-l** or **-s** to specify the desired partition.
-o	Open application access.
-p <password>	Specifies the challenge password - if a challenge password exists and this argument is not included, login will not be performed.
-q <port>	Specifies the remote PEDserver port. **Default:** 1503
-r <server_IP>	Specifies the remote PEDserver IP.
-s <slot>	Specifies the slot ID number. Include either **-l** or **-s** to specify the desired partition. **Default:** 0

Argument(s)	Description
-u	Specifies that login should be performed as the Crypto User. If this argument is not included, the Crypto Officer will be logged in.
-v	Show verbose logs.

Examples

```
salogin -o -s 1 -i 1:1
# open a persistent application connection
# on slot 1 with app id 1:1


salogin -o -s 1 -i 1:1 -p HT7bHTHPRp/4/Cdb
# open a persistent application connection
# and login with Luna HSM challenge


salogin -c -s 1 -i 1:1
# close persistent application connection 1:1
# on slot 1
```

Attempting to use salogin on an STC-enabled slot

```
lunacm:>slot list

Slot Id ->              0
Label ->                stc_ppso
Serial Number ->        1213429268189
Model ->                LunaSA
Firmware Version ->     7.0.1
Configuration ->        Luna User Partition With SO (PW) Signing With Cloning Mode
Slot Description ->     Net Token Slot

Current Slot Id: 0

Command Result : No Error



lunacm:>stc status

Enabled:            Yes
Status:             Connected
Channel ID:         3
Cipher Name:        AES 256 Bit with Cipher Block Chaining
HMAC Name:          HMAC with SHA 512 Bit

Command Result : No Error



lunacm:>stc identityshow

Client Identity Name:          mySTCclientID
```

```
Public Key SHA1 Hash:          58feec48e485762c39a8c32f94cf535bf545699e
List of Registered Partitions:

Partition Identity   Partition        Partition Public Key SHA1 Hash
Label                Serial Number
_____________________________________________________________________________

par1                 1213429268189    d4d4d65d281fd17580c56ddf09439c79c466a09a

Command Result : No Error



lunacm:>clientconfig listservers

Server ID  Server                          Channel
_____________________________________________________

0          192.20.11.184                   STC

Command Result : No Error

lunacm:>exit

# ./salogin -o -s 0 -i 1:1 -p userpin
CA_OpenApplicationID: failed to open application id. err 0x80000030
token not present or app id already open?
```

Other Options

For Java applications, consider using the KeyStore interface. It is internally consistent with the service provider interface defined by SUN/Oracle and does not require any proprietary code or applications.

> **NOTE** The Luna Keystore is not a physical file like a regular JKS. It is a virtual interface to the HSM and contains only handles for the private key objects.

If you are using an integration that does not refer to a KeyStore then the **salogin** utility might be required. You are then limited to working with one partition. The utility will work with any Luna HSM, as long as it is visible to the client at the time the library is initialized.

CHAPTER 6: pscp

The Luna HSM Client software includes the **pscp** utility, used to securely move updates and certificates and other files from a source computer onto the Luna Network HSM appliance, or to move appliance certificates or log files out to a client computer.

> **NOTE** For Linux/UNIX-based operating systems, you can also use the standard **scp** utility, with the same syntax described for **pscp**.

All packages from Thales are signed and encrypted and come with an authorization code (authcode) that must be provided to decrypt and use the package.

Syntax

Client to appliance
pscp [options] [<user>@]<host/IP>:<source> <target>

Appliance to client
pscp [options] <source> [<source>...] [<user>@]<host/IP>:<target>

List files on the appliance
pscp [options] **-ls** <user>@<host/IP>:<file_path>

> **NOTE** When using **scp** or **pscp** over an IPv6 network, enclose addresses in square brackets.

Argument(s)	Description
-p	Preserve file attributes.
-q	Quiet -- do not show statistics.
-r	Copy directories recursively.
-S <path_to_SSH>	Specify the location of SSH.
-v	Show verbose messages.
-P <port>	Connect to the specified port.
-pw <password>	Login with specified password.

Argument(s)	Description
-unsafe	Allow server-side wildcards (dangerous).

Examples

The following examples illustrate how to transfer files from a Luna HSM Client to a Luna Network HSM, and from a Luna Network HSM to a Luna HSM Client.

Transferring a file from a Luna HSM Client to a Luna Network HSM

```
/usr/safenet/lunaclient/bin/>scp test-file.txt admin@myluna:
admin@myluna's password: ********
test-file.txt             |           0  kB |   0.1  kB/s | ETA: 00:00:00 | 100%

/usr/safenet/lunaclient/bin/>
```

The colon is required. Type nothing after the colon when moving files onto the Luna Network HSM appliance. All files that are **pscp**'d to the appliance go to a predetermined directory, which you cannot change (for security reasons). While it is possible to change the filename during pscp (by typing a new filename after the colon in the pscp command), this is not recommended, since most operations expect certain filenames and can fail if those are not found.

```
/usr/safenet/lunaclient/bin/>pscp test-file.txt admin@myluna:different-file.txt
admin@myluna's password: ********
test-file.txt             |           0  kB |   0.1  kB/s | ETA: 00:00:00 | 100%

/usr/safenet/lunaclient/>
```

If the arriving file carries an unexpected name, it might not be handled correctly by subsequent commands.

If you have SSH located in a non-standard (UNIX) location, launch the **pscp** command with the "-S" option (that's an uppercase "S"), followed by the path to SSH, before supplying the paths to the source and target files, like:

```
pscp -S /usr/bin/ssh <source file> <dest file>
```

Transferring a file from a Luna Network HSM to a Luna HSM Client

```
bash-2.05# pscp admin@myLuna3:server.pem .
admin@myLuna3's password: *********
server.pem          100%
|**************************************************|   928
00:0
```

Note the dot (.) at the end of the command, denoting “place the resulting file in the current directory”.

CHAPTER 7: vtl

The **vtl** (Virtual Token Library) command-line utility is installed with the Luna HSM Client software. It is used to manage the relationship between your Client computer and one or more Luna appliances.

> **NOTE** Many **vtl** functions have been moved to LunaCM. Thales recommends using LunaCM for client configuration wherever possible. See lunacm:> **clientconfig** for details.

Open a command prompt window or console, cd to the directory where you installed your client software, and run the **vtl** command (with the **-h** option, to see the available sub-commands).

These are the commands that you can use to manage the relationship between your Luna HSM Client computer and one or more Luna appliances (either Luna Network HSMs, or Luna Backup HSM configured for remote backup). You must have Administrator privileges on the client computer. If you do not also have authority on the Luna Network HSM appliance(s), then you need the co-operation of the person who holds that authority.

```
admin@mycomputer:~>vtl
usage: (select command -h for additional information)
```

> **NOTE** You need to be Administrator (or equivalent) when running **vtl** commands that need to access /etc and /user (and the equivalents in Windows).

Subcommands

Subcommand	Description
addCA	Add a Certificate Authority root chain certificate to the list of CAs registered on the client. See "vtl addCA" on page 92.
addServer	Adds the specified server to the client's list of trusted servers. See "vtl addServer" on page 93.
addServerNoCert	Add an HSM server's IP/hostname to the client's list of Luna Network HSM servers. See "vtl addServerNoCert" on page 94.
cklogsupport	Enable or disable CKLOG support. See "vtl cklogsupport" on page 95.
createCert	Create (or re-create) the client's certificate and private key used for NTLS (Network Trust Link Service). See "vtl createCert" on page 96.
createCSR	Create a Certificate Signing Request (CSR)—a private key and unsigned client certificate. See "vtl createCSR" on page 98.

Subcommand	Description
deleteCA	Delete a Certificate Authority root chain certificate from the truststore on the client. See "vtl deleteCA" on page 100.
deleteServer	Remove a server/host from the client's list of trusted HSM servers. See "vtl deleteServer" on page 101.
deleteServerNoCert	Delete the IP/hostname of a Luna Network HSM server from the list of servers, without deleting the certificate associated with that server. See "vtl deleteServerNoCert" on page 102.
examineCert	Display details of a specified certificate. See "vtl examineCert" on page 103.
fingerprint	Display the fingerprint of a specified certificate. See "vtl fingerprint" on page 105.
listCAs	Display a list of the Certificate Authority root chain certificates registered on the client. See "vtl listCAs" on page 106.
listServers	Display a list of HSM servers trusted by this client. See "vtl listServers" on page 107.
listSlots	List all PKCS#11 cryptographic device slots that can be seen at this time. See "vtl listSlots" on page 108.
logging	Configure logging for Windows computers. See "vtl logging" on page 109.
replaceServer	Replace a named server/host from the client's list of trusted HSM servers with a new named server/host. See" vtl replaceServer" on page 110.
supportInfo	Create a support information file, when one is requested by Thales Customer Support. See "vtl supportInfo" on page 111.
verify	Verify the visible HSM slots or partitions. See "vtl verify" on page 112.

vtl addCA

Add a Certificate Authority root chain certificate to the client's trust store. This will allow the client to connect to partitions on a Luna Network HSM whose certificate is signed by the same CA.

You must be Administrator on your Client computer, or logged in as a user with Administrator privileges.

> **NOTE** This feature requires minimum Luna HSM Client 10.1.0.

Syntax

vtl addCA -n <CA_name> **-c** <cert_filepath/name>

Argument(s)	Description
-n <IP/hostname>	Name used to identify the Certificate Authority.
-c <cert_filepath/name>	The name (including the path to its location on your computer) of the CA's certificate file.

Example

```
>vtl addCA -n CAroot -c "C:\Program Files\SafeNet\LunaClient\CAroot.pem"
vtl (64-bit) v10.1.0. Copyright (c) 2019 SafeNet. All rights reserved.


New server CAroot successfully added to server list.
```

vtl addServer

Adds the specified server to the client's list of trusted servers. You may wish to check the fingerprint of the server certificate with the "vtl fingerprint" on page 105 command before adding it. The server certificate is one that you have imported from the Luna Network HSM appliance to your Client computer, using **"pscp" on page 88** or **scp**.

You must be Administrator on your Client computer, or logged in as a user with Administrator privileges.

Syntax

vtl addServer -n <IP/hostname> **-c** <cert_filename>

Argument(s)	Description
-n <IP/hostname>	The hostname or IP address of the server to add. Use the IP address if the server's certificate uses its IP address instead of its hostname. If you are uncertain what format the server's certificate uses, contact your Luna Network HSM appliance administrator, or look for the “CN=” field when using the "vtl examineCert" on page 103 command.
-c <cert_filename>	The name (including the path to its location on your computer) of the server's certificate file. Use the **"pscp" on page 88** or **scp** utility to collect the server's certificate from the appliance, or use the certificate provided by your Luna Network HSM appliance administrator. You may wish to confirm the authenticity of the certificate by using "vtl fingerprint" on page 105.

Example

```
$ ./vtl add -n 192.20.9.161 -c server161.pem
New server 192.20.9.161 successfully added to server list..
```

vtl addServerNoCert

Add an HSM server's IP/hostname to the client's list of Luna Network HSM servers. You must also add the Certificate Authority's certificate chain to the client's trust store to establish a trusted connection (see "vtl addCA" on page 92).

You must be Administrator on your Client computer, or logged in as a user with Administrator privileges.

> **NOTE** This feature requires minimum Luna HSM Client 10.1.0.

Syntax

vtl addServerNoCert-n <IP/hostname>

Argument(s)	Description
-n <IP/hostname>	The hostname or IP address of the server to add. Use the IP address if the server's certificate uses its IP address instead of its hostname. If you are uncertain what format the server's certificate uses, contact your Luna Network HSM appliance administrator.

Example

```
>vtl addservernocert -n 192.168.11.10
vtl (64-bit) v10.1.0. Copyright (c) 2019 SafeNet. All rights reserved.


New server 192.168.11.10 successfully added to server list.
```

vtl cklogsupport

Enable or disable CKLOG support. CKLOG is a facility which can record all interactions between an application and our PKCS#11-compliant library. It allows a developer to debug an application by viewing what the library receives. See Libraries and Applications for more information.

Syntax

vtl cklogsupport {**enable** | **disable**}

Example

```
$ ./vtl cklogsupport enable
Chrysoki2 LibUNIX = /usr/lib/libCryptoki2.so
Cklog not enabled
Enabling cklog
```

vtl createCert

Creates the client's certificate and private key that are used by NTLS. Re-creates the key and certificate if they already exist.

> **CAUTION!** If the key and certificate are re-created, the client will need to be removed and re-registered on each of the HSM servers with which it was registered.

> **NOTE** The client hostname/IP (**-n**) is the only mandatory field for certificate creation. All other fields of the certificate are used simply for display and visual confirmation purposes. The NTLA never displays certificate data fields to the user, so the content in these fields is irrelevant.

Syntax

vtl createCert -n <IP/hostname> [**-c** <country_code>] [**-s** <state>] [**-l** <locality>] [**-o** <organization>] [**-u** <organization_unit>] [**-e** <email_address>] [**-P** <private_key_filename>] [**-C** <cert_filename>] [**-d** <certificate_validity_period>] [**-v**]

Argument(s)	Description
-c <country>	The country where the client computer resides.
-C <filename>	The specified filename (***.pem**) for the certificate. Default: <IP/hostname>**.pem** **NOTE** Thales recommends using the default filename to avoid losing track of keys and certificates.
-d <validity_period>	Specifies the validity period for the client certificate, in days. Default: **3650** (10 years)
-e <email_adress>	An email address to contact the certificate creator.
-l <locality>	The locality where the client computer resides.
-n <IP/hostname>	The client hostname or IP address. This becomes the certificate Common Name (CN).
-o <organization>	The name of the organization that owns the client computer.
-P <filename>	The specified filename (***Key.pem**) for the private key. Default: <IP/hostname>**Key.pem** **NOTE** Thales recommends using the default filename to avoid losing track of keys and certificates.

Argument(s)	Description
-s <state>	The state where the client computer resides.
-u <unit>	The business unit or department that owns the client computer.
-v	Verbose mode. Output extra information while creating the certificate and private key.
-x	Deprecated option to encrypt the private key -- the private key is always encrypted by default.

Example

Windows

```
vtl createCert -n test
Private Key created and written to: E:\temp\clientCerts\testKey.pem
Certificate created and written to: E:\temp\clientCerts\test.pem

vtl createCert -n test -v
Using configuration from C:\Program Files\SafeNet\LunaClient\openssl.cnf
It needs to be at least 1024
Writing new private key to stdout E:\temp\clientCerts\testKey.pem'
CA [CA]:CA
Ontario [Ontario]:Ontario
Ottawa [Ottawa]:Ottawa
My company [My company]:My company
 []:
test [test]:test
 []:
Private Key created and written to: E:\temp\clientCerts\testKey.pem
Certificate created and written to: E:\temp\clientCerts\test.pem
```

UNIX

```
vtl createCert -n test
Private Key created and written to: /usr/safenet/lunaclient/cert/client/testKey.pem
Certificate created and written to: /usr/safenet/lunaclient/cert/client/test.pem
```


vtl createCSR

Create a Certificate Signing Request (CSR)—a private key and unsigned client certificate. The certificate must be signed by a third party before being used to authenticate the Luna HSM Client.

> **CAUTION!** If the key and certificate are re-created, existing NTLS connections are broken and the client must be removed and re-registered on each HSM server.

> **NOTE** The client hostname/IP (**-n**) is the only mandatory field for certificate creation. All other fields of the certificate are used simply for display and visual confirmation purposes. The NTLA never displays certificate data fields to the user, so the content in these fields is irrelevant.
>
> This feature requires minimum Luna HSM Client 10.1.0.

Syntax

vtl createCSR -n <IP/hostname> [**-c** <country_code>] [**-s** <state>] [**-l** <locality>] [**-o** <organization>] [**-u** <organization_unit>] [**-e** <email_address>] [**-P** <private_key_filename>] [**-C** <cert_filename>] [**-d** <certificate_validity_period>] [**-v**]

Argument(s)	Description
-c <country>	The country where the client computer resides.
-C <filename>	The specified filename (***CSR.pem**) for the unsigned certificate. Default: <IP/hostname>**CSR.pem** **NOTE** Thales recommends using the default filename to avoid losing track of keys and certificates.
-d <validity_period>	Specifies the validity period for the client certificate, in days. Default: **3650** (10 years)
-e <email_address>	An email address to contact the certificate creator.
-l <locality>	The locality where the client computer resides.
-n <IP/hostname>	The client hostname or IP address. This becomes the certificate Common Name (CN).
-o <organization>	The name of the organization that owns the client computer.
-P <filename>	The specified filename (***Key.pem**) for the private key. Default: <IP/hostname>**Key.pem** **NOTE** Thales recommends using the default filename to avoid losing track of keys and certificates.

Argument(s)	Description
-s <state>	The state where the client computer resides.
-u <unit>	The business unit or department that owns the client computer.
-v	Verbose mode. Output extra information while creating the certificate and private key.
-x	Deprecated option to encrypt the private key -- the private key is always encrypted by default.

Example

```
>vtl createCSR -n 192.168.10.12
vtl (64-bit) v10.1.0. Copyright (c) 2019 SafeNet. All rights reserved.

Private Key created and written to: C:\Program
Files\SafeNet\LunaClient\cert\client\192.168.10.12Key.pem
Certificate CSR created and written to: C:\Program
Files\SafeNet\LunaClient\cert\client\192.168.10.12CSR.pem
```

vtl deleteCA

Delete a Certificate Authority root chain certificate from the trust store on the client. This will break the NTLS connection between this client and any Luna Network HSM appliance authenticated by this CA.

You must be Administrator on your Client computer, or logged in as a user with Administrator privileges.

> **NOTE** This feature requires minimum Luna HSM Client 10.1.0.

Syntax

vtl deleteCA -n <cert_name>

Argument(s)	Description
-n <cert_name>	The name of the certificate to be deleted.

Example

```
>vtl deleteCA -n CAroot
vtl (64-bit) v10.1.0. Copyright (c) 2019 SafeNet. All rights reserved.


CA CAroot successfully removed from server/CA list.
```

vtl deleteServer

Removes the given host from the list of trusted HSM servers. View a list of all trusted servers with the command "vtl listServers" on page 107.

Syntax

vtl deleteServer -n <IP/hostname>

Argument(s)	Description
-n <IP/hostname>	The hostname or IP address of the HSM server to delete.

Example

```
vtl delete -n LunaSA1
Server lunasa1 successfully removed from server list.
```


vtl deleteServerNoCert

Delete the IP/hostname of a Luna Network HSM server from the list of servers, without deleting the certificate associated with that server. Use "vtl deleteCA" on page 100 to remove a certificate from the trust store.

> **NOTE** This feature requires minimum Luna HSM Client 10.1.0.

Syntax

vtl deleteServerNoCert -n <IP/hostname>

Argument(s)	Description
-n <IP/hostname>	The hostname or IP address of the Luna Network HSM server to delete.

Example

```
>vtl deleteservernocert -n 192.168.11.10
vtl (64-bit) v10.1.0. Copyright (c) 2019 SafeNet. All rights reserved.


Server 192.168.11.10 successfully removed from server/CA list.
```

vtl examineCert

Displays the details of the specified certificate. If the command is issued with no additional parameters, it returns the client certificate. If the **-f** option is used, then a filespec is required, and the command returns the details of the indicated certificate.

Syntax

vtl examineCert [**-f** <filespec>]

Argument(s)	Description
-f <filespec>	Specify the filespec of the certificate to return details for. The server cert files are located in the cert/server directory (<name>**Cert.pem**, where <name> is the name specified when the server was added with "vtl addServer" on page 93.

Example

Windows

```
C:\Program Files\SafeNet\LunaClient>vtl examineCert -f cert\server\bigCert.pem
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 0 (0x0)
Signature Algorithm: sha256WithRSAEncryption
Issuer: C=CA, ST=Ontario, L=Ottawa, O=Chrysalis-ITS, CN=168.0.1.0
Validity
  Not Before: Nov 10 14:10:36 2011 GMT
  Not After : Nov 11 14:10:36 2021 GMT
Subject: C=CA, ST=Ontario, L=Ottawa, O=Chrysalis-ITS, CN=168.0.1.0
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  RSA Public Key: (2048 bit)
    Modulus (2048 bit):
        00:a9:c3:db:59:33:b8:65:20:c9:13:f7:a7:e5:59:
      7b:12:a4:31:d3:62:36:9a:62:68:6e:1d:d7:c7:f0:
      8c:fd:06:43:f8:42:f7:8c:de:74:d1:38:a3:8f:37:
      94:c4:82:cc:67:d8:51:14:cd:e4:b7:dd:f8:ff:09:
      c8:03:f9:62:c5:ad:fc:4d:2e:fe:67:dd:6b:e7:de:
      bd:9e:bd:92:14:63:a6:99:2a:78:e7:72:6d:ba:79:
      3d:55:a8:a4:5d:85:11:36:9f:3d:4c:9a:e6:e8:bf:
      b4:5b:45:83:46:c4:2c:d9:22:fa:50:5a:28:ba:6e:
      2f:cb:2f:54:47:8d:3b:fd:73:bc:5a:ce:cd:bb:4e:
      ec:b5:1c:87:b6:b1:cd:53:77:f0:f2:36:e9:b2:3d:
      2e:61:6f:f2:73:c6:ad:c4:d4:fe:20:3b:de:e8:a9:
      a4:cd:93:17:0a:65:a5:58:ef:e3:11:d5:f0:ac:92:
      af:33:dc:1c:c0:8f:04:fc:13:53:65:7f:52:34:07:
      71:7a:9b:e5:d8:1e:e0:bd:ca:13:0f:f9:00:33:e5:
      2a:0c:79:78:42:ff:4c:1a:d6:83:2c:ae:bf:2d:1d:
      93:ac:f5:6b:60:97:ab:fb:1a:d5:86:2c:2f:3c:f6:
      7e:37:8d:77:0a:7a:dd:7c:38:61:26:9a:c9:c0:0d:
      b3:57
```



```
Exponent: 65537 (0x10001)
Signature Algorithm: sha256WithRSAEncryption
15:49:31:22:c4:1a:80:9f:2d:de:4b:df:63:b8:b0:16:b0:af:
7a:f4:8f:62:0b:ad:fa:21:b5:95:6e:fc:a6:09:b9:f9:5f:ea:
8e:c8:a7:d5:90:0b:12:ff:a6:34:b5:9a:02:7f:81:66:38:21:
c7:92:21:a2:d4:0f:e9:44:84:2a:f5:ea:d2:00:4b:f1:0f:d5:
55:5b:15:3e:b4:b5:b6:d4:32:7d:fe:8c:ef:80:ef:f8:dd:73:
e6:1e:a2:41:4c:8c:1d:c7:fa:2a:a9:25:ef:aa:29:8e:40:8e:
da:2a:3d:af:67:a7:7e:da:a9:76:6d:c6:10:e7:3a:5d:45:ac:
a0:f3:35:30:44:76:7c:b0:ce:61:19:0b:74:b1:3f:51:08:f9:
12:47:75:7c:33:0c:ee:02:d7:bb:48:10:6d:40:5b:fe:26:f2:
8f:28:0f:d9:2d:25:d9:af:49:44:b3:25:c6:cf:97:21:f0:3a:
0d:0e:41:30:34:56:e8:8d:6b:d6:36:fb:a9:79:e6:bc:dd:6b:
61:cf:98:01:c0:70:b2:81:41:1c:79:6e:58:47:e9:22:83:98:
9f:9f:62:87:e3:74:df:87:fe:0b:78:55:0f:1e:6e:56:21:b6:
0e:29:64:cb:75:de:90:82:bd:24:64:ef:db:8c:9b:5b:b4:7e:
86:61:89:64
```

The only difference for a UNIX client would be the path in the filespec.

vtl fingerprint

Displays the fingerprint of the specified certificate. If the command is issued with no additional parameters, it returns the client fingerprint. If the **-f** option is used, then a filespec is required, and the command returns the fingerprint of the indicated certificate.

Syntax

vtl fingerprint [**-f** <filespec>]

Argument(s)	Description
-f <filespec>	Specify the filespec of the certificate to return details for. The server cert files are located in the cert/server directory (<name>**Cert.pem**, where <name> is the name specified when the server was added with "vtl addServer" on page 93.

Example

```
vtl fingerprint
Certificate fingerprint: 91:01:EC:BA:6A:31:19:69:CF:8D:1A:23:87:95:76:35.
```

vtl listCAs

Display a list of the Certificate Authority certificate chains registered in the client's trust store.

> **NOTE** This feature requires minimum Luna HSM Client 10.1.0.

Syntax

vtl listCAs

Example

```
>vtl listCAs
vtl (64-bit) v10.1.0. Copyright (c) 2019 SafeNet. All rights reserved.

CAroot
  subject= /CN=OTT1-TITAN-CA
  issuer= /CN=OTT1-TITAN-CA
```

vtl listServers

Displays a list of the HSM servers trusted by this client.

Syntax

vtl listServers

Example

```
>vtl listservers

Server: 192.20.10.10
Server: testserver
```


vtl listSlots

Displays a list of all slots found.

Syntax

vtl listSlots

Example

```
>vtl listSlots
Number of slots: 3
The following slots were found:


Slot#   Description          Label                    Serial#         Status
======  ===============      ==================       ========        ===========
     0  Net Token Slot       kbPSO                    1311583664227   Present
     1  User Token Slot      mypciepsopar             349297122736    Present
     2  Admin Token Slot     mypcie6                  150022          Present
     3  Net Token Slot       dpod-service             12345678        Present
     4  Luna UHD Slot        myUSBpw                  7001312         Present
     5  Luna UHD Slot        -                        -               Not present
     6  Net Admin Token Slot myRBSG5Bk                7000329         Present
```

NOTE In the example list above:

- slot 0 represents a network-linked application partition on a Luna Network HSM
- slot 1 is the application partition on a Luna PCIe HSM
- slot 2 is the HSM administrative partition of the same Luna PCIe HSM
- slot 3 is a Luna Cloud HSM Service available through Data Protection on Demand.
- slot 4 is a Luna USB HSM 7
- slot 5 is a placeholder slot for an HSM that could be attached to a USB port
- slot 6 is the HSM administrative partition of a Luna Backup HSM that is connected to this client via Remote Backup Service

You won't necessarily see all, or even most of those in your situation, with your equipment; the list in the example merely shows how different types are presented. Luna PCIe HSM and Luna Cloud HSM service slots can display with this command, but can only be managed through LunaCM.

vtl logging

Configure the directory path where log files are to be stored.

The client library writes log messages to SYSLOG on Linux/UNIX systems. However, for Windows, the log messages are written to the file "LunaCryptokiLog.htm" at the location that you specify in <logPath>.

Syntax

vtl logging

configure <log_path>
show

Argument(s)	Description
configure <log_path>	Specify the directory path where log files are to be stored.
show	Displays the current directory path where log files are stored.

To demonstrate that the logging is working on a Windows platform, you could create an error situation as follows:

1. Enable the client side log on a Windows platform.
2. Create a client certificate.
3. Register the client with a Luna Network HSM appliance.
4. Manually delete the client certificate file.
5. Run **ckdemo** or another application against a partition on that Luna Network HSM. NTLS is broken for this client (due to the missing certificate), so any commands from your application should fail.
6. Check LunaCryptokiLog.htm and observe error messages written there.

Examples

```
C:\Program Files\SafeNet\LunaClient>vtl logging configure "C:\Program Files\SafeNet\LunaClient"
Success setting log path to C:\Program Files\SafeNet\LunaClient
C:\Program Files\SafeNet\LunaClient>vtl logging show
Client logging written to: C:\Program Files\SafeNet\LunaClient\LunaCryptokiLog.htm
```


vtl replaceServer

Replaces the specified old server in the client's list of trusted HSM servers, with the specified new server.

Syntax

vtl replaceServer -o <old_hostname/IP> **-n** <new_hostname/IP> **-c** <certificate_file>

Argument(s)	Description
-c <certificate_file>	The name and filepath of the HSM server's certificate file.
-n <new_hostname/IP>	The hostname or IP address of the server that is replacing the old server. Use the IP address if the server's certificate uses its IP address instead of its hostname. If you are uncertain what format the server's certificate uses, contact your Luna Network HSM appliance administrator or look for the “CN=” field in the output from "vtl examineCert" on page 103.
-o <old_hostname/IP>	The hostname or IP address of the Luna Network HSM server being replaced. Use the IP address if the server's certificate uses its IP address instead of its hostname.

Example

```
bash # ./vtl replaceServer -o yourluna -n myluna -c server.pem
New server myluna successfully added to server list.
Server yourluna successfully replaced with myluna.
```

vtl supportInfo

Creates a client-side support information file (may be requested by Thales Technical Support to help resolve an issue).

Syntax

vtl supportInfo

Example

```
>vtl supportinfo


Creating client-side support information file now...

'vtl supportInfo' completed. File "c_supportInfo.txt" created.
```


vtl verify

Verify the Luna Network HSM slots/partitions visible to this Client.

Syntax

vtl verify

Example

```
bash-2.03# ./vtl verify

The following Luna Network HSM Slots/Partitions were found:
Slot    Serial #                Label
====    ================        =====
   1            65091001        MyPartition
   2            65097001        YourPartition
   3            65093001        HisPartition
```

