YubiHSM 2 Commands

Yubico

May 11, 2022

CONTENTS

CHAPTER

ONE

INTRODUCTION

This section contains a list of the commands supported by the YubiHSM 2.

The low-level format for each command message and the relative response is provided, together with an example of how that command can be used within the yubihsm-shell.

The numerical codes corresponding to each command are provided below:

Command Name	Hex Value
Echo	0x01
Create Session	0x03
Authenticate Session	0x04
Session Message	0x05
Get Device Info	0x06
Reset Device	0x08
Close Session	0x40
Get Storage Info	0x41
Put Opaque	0x42
Get Opaque	0x43
Put Authentication Key	0x44
Put Asymmetric Key	0x45
Generate Asymmetric Key	0x46
Sign Pkcs1	0x47
List Objects	0x48
Decrypt Pkcs1	0x49
Export Wrapped	0x4a
Import Wrapped	0x4b
Put Wrap Key	0x4c
Get Log Entries	0x4d
Get Object Info	0x4e
Set Option	0x4f
Get Option	0x50
Get Pseudo Random	0x51
Put Hmac Key	0x52
Sign Hmac	0x53
Get Public Key	0x54
Sign Pss	0x55
Sign Ecdsa	0x56
Derive Ecdh	0x57
Delete Object	0x58
Decrypt Oaep	0x59

continues on next page

Table 1 – continued from previous page

Command Name	Hex Value
Generate Hmac Key	0x5a
Generate Wrap Key	0x5b
Verify Hmac	0x5c
Sign Ssh Certificate	0x5d
Put Template	0x5e
Get Template	0x5f
Decrypt Otp	0x60
Create Otp Aead	0x61
Randomize Otp Aead	0x62
Rewrap Otp Aead	0x63
Sign Attestation Certificate	0x64
Put Otp Aead Key	0x65
Generate Otp Aead Key	0x66
Set Log Index	0x67
Wrap Data	0x68
Unwrap Data	0x69
Sign Eddsa	0x6a
Blink Device	0x6b
Change Authentication Key	0x6c

CHAPTER

TWO

AUTHENTICATE SESSION COMMAND

Complete the mutual authentication process started with *CREATE SESSION Command*.

2.1 Description

Finish the Session negotiation and authenticate the Session to the device. After this command completes successfully the Session is authenticated and can be used.

2.2 Shell Example

Create a new Session with Authentication Key 1 using the password `password`, this performs both the creation and authentication steps:

```
yubihsm> session open 1 password
Created session 0
```

2.3 Protocol Details

2.3.1 Command

Tc = 0x04
Lc = 17
Vc = S || B || M

Parameters:

S := Session ID (1 byte)

B := Host Cryptogram (8 bytes)

M := CMAC(S-MAC, 016 || T || Lc + 8 || S || B) (8 bytes)

This is the first authenticated message in the chain. The device verifies M and B, both using S-MAC.

2.3.2 Response

Tr = 0x84
Lr = 0
Vr = Ø

CHAPTER

THREE

BLINK DEVICE COMMAND

Blink the device.

3.1 Description

Blink the LED of the device to identify it.

3.2 Shell Example

Blink the device for 15 seconds:

```
yubihsm> blink 0 15
```

3.3 Protocol Details

3.3.1 Command

Tc = 0x6b
Lc = 1
Vc = S

Parameters:

S := Seconds to blink for (1 byte)

3.3.2 Response

Tr = 0xeb
Lr = 0
Vr = Ø

CHAPTER

FOUR

CHANGE AUTHENTICATION KEY COMMAND

Change an Authentication Key.

4.1 Description

Replace the Authentication Key used to establish the current Session. It is not possible to modify any of the metadata connected to the Object such as Domains or Capabilities. Only the payload data of the Object (i.e., the long-lived symmetric keys) will be modified.

The same PBKDF2 derivation scheme described in Session is available.

4.2 Shell Example

Change the current Authentication Key deriving it from the password `newpassword`:

```
yubihsm> change authkey 0 1 newpassword
Changed Authentication key 0x0001
```

4.3 Protocol Details

4.3.1 Command

Tc = 0x6c
Lc = 2 + 1 + 16 + 16
Vc = I || A || Ke || Km

Replace the currently used Authentication Key with a new set of keys.

Parameters:

`I` := Object ID of the Authentication Key (2 bytes)

`A` := Algorithm (1 byte)

`Ke` := Encryption Key (16 bytes)

`Km` := Mac Key (16 bytes)

4.3.2 Response

Tr = 0xec
Lr = 2
Vr = I

Parameters:

I := Object ID of the changed Object (2 bytes)

CHAPTER

FIVE

CLOSE SESSION COMMAND

Close session.

5.1 Description

Close the current Session and release it for re-use.

5.2 Shell Example

Close Session 0:

```
yubihsm> session close 0
```

5.3 Protocol Details

5.3.1 Command

Tc = 0x40
Lc = 0
Vc = Ø

5.3.2 Response

Tr = 0xc0
Lr = 0
Vr = Ø

CHAPTER

SIX

CREATE OTP AEAD COMMAND

Create a Yubico OTP AEAD.

6.1 Description

Create a Yubico OTP AEAD using the provided data.

6.2 Shell Example

Create a new AEAD using Otp-aead Key `0x027c` with the key `000102030405060708090a0b0c0d0e0f` and private ID `010203040506`. Store the result in the file `aead`:

```
yubihsm> otp aead_create 0 0x027c 000102030405060708090a0b0c0d0e0f 010203040506 aead
```

6.3 Protocol Details

6.3.1 Command

Tc = 0x61
Lc = 24
Vc = I || K || P

Parameters:

`I` := Object ID of the OTP AEAD Key (2 bytes)

`K` := OTP Key (16 bytes)

`P` := OTP Private ID (6 bytes)

6.3.2 Response

Tr = 0xe1
Lr = LA
Vr = A

Parameters:

A := Nonce concatenated with AEAD (36 bytes)

CHAPTER

SEVEN

CREATE SESSION COMMAND

Begin the mutual authentication process for establishing a Session.

7.1 Description

Start negotiating a Session with the device. This command tells the device which Authentication Key to use and sends the host challenge part. The response will contain the device challenge and device authentication part. To establish the session continue with *AUTHENTICATE SESSION Command*.

7.2 Shell Example

Create a new session with Authentication Key 1 using the password `password`. This does both the session creation and authentication steps:

```
yubihsm> session open 1 password
Created session 0
```

7.3 Protocol Details

7.3.1 Command

Tc = 0x03
Lc = 10
Vc = I || H

Parameters:

`I` := Key set ID (2 bytes)

`H` := Host Challenge (8 bytes)

The device generates a random Card Challenge `C` (8 bytes).

The device derives three Session Keys (`S-ENC`, `S-MAC` and `S-RMAC`) starting from the set of two static keys identified by `I` (`K-ENC` and `K-MAC`) and the two challenges `H` and `C`, using the same procedure described in SCP03.

The device uses `S-MAC` together with `H` and `C` to compute the Card Cryptogram `A`. The host will compute the Host Cryptogram `B` after having received `C` and derived `S-MAC`.

On success the device generates a Session ID S (1 byte) and sets the message counter for the current Session to 1.

7.3.2 Response

Tr = 0x83
Lr = 17
Vr = S || C || A

CHAPTER

EIGHT

DECRYPT OAEP COMMAND

Decrypt data using RSA-OAEP.

8.1 Description

Decrypt data encrypted with RSA-OAEP

8.2 Shell Example

Decrypt data stored in file enc using key `0x79c3`:

```
yubihsm> decrypt oaep 0 0x79c3 rsa-oaep-sha1 enc
xlwIc7yQf/KkV5v4Y87Q9ZSqLReoNAxlCmmMPA4W08U=
```

8.3 Protocol Details

8.3.1 Command

`Tc = 0x59`
`Lc = 2 + 1 + LD + LH`
`Vc = I || M || D || Hl`

Parameters:

`I` := Object ID of the Asymmetric Key (2 bytes)

`M` := Hash Algorithm to use for MGF1 (1 byte)

`D` := Decryption data (256, 384 or 512 bytes)

`Hl` := Hash of OAEP Label (20, 32, 48 or 64 bytes)

8.3.2 Response

Tr = 0xc9
Lr = LR
Vr = R

Parameters:

R := Decrypted data with OAEP padding removed

CHAPTER

NINE

DECRYPT OTP COMMAND

Decrypt a Yubico OTP.

9.1 Description

Decrypt a Yubico OTP and return counters and timer information.

9.2 Shell Example

Decrypt a (hex encoded) Yubico OTP using key ID `0x027c`:

```
yubihsm> otp decrypt 0 0x027c 2f5d71a4915dec304aa13ccf97bb0dbb aead
   OTP decoded, useCtr:1, sessionCtr:1, tstph:1, tstpl:1
```

9.3 Protocol Details

9.3.1 Command

Tc = 0x60
Lc = 2 + 36 + 16
Vc = K || A || O

Parameters:

`I` := Object ID of the OTP AEAD Key (2 bytes)

`A` := Nonce concatenated with AEAD (36 bytes)

`O` := OTP (16 bytes)

9.3.2 Response

Tr = 0xe0
Lr = 6
Vr = S || U || Th || Tl

Parameters:

S := Session counter (2 bytes)

U := Usage counter (1 byte)

Th := Timestamp high (1 byte)

Tl := Timestamp low (2 bytes)

CHAPTER

TEN

DECRYPT PKCS1 COMMAND

Decrypt data that was encrypted using RSA-PKCS#1v1.5.

10.1 Description

Decrypt data encrypted with RSA-PKCS#1v1.5

10.2 Shell Example

Decrypt the file `enc` using key `0xa930`:

```
yubihsm> decrypt pkcs1v1_5 0 0xa930 enc
   xlwIc7yQf/KkV5v4Y87Q9ZSqLReoNAxlCmmMPA4W08U=
```

10.3 Protocol Details

10.3.1 Command

`Tc = 0x49`
`Lc = 2 + LD`
`Vc = I || D`

Parameters:

`I` := Object ID of the Asymmetric Key (2 bytes)

`D` := Decryption data (256, 384 or 512 bytes)

The data is padded using the PKCS#1v1.5 scheme with `Block Type 2`. The data is decrypted and conformance to the padding scheme must be checked. Padding is then removed and the contained message is returned.

10.3.2 Response

Tr = 0xc9
Lr = LR
Vr = R

Paramters:

R := Decrypted data with padding removed

CHAPTER

ELEVEN

DELETE OBJECT COMMAND

Delete an Object.

11.1 Description

Delete an Object in the device.

11.2 Shell Example

Delete Asymmetric Key `0x52b6`:

```
yubihsm> delete 0 0x52b6 asymmetric-key
```

11.3 Protocol Details

11.3.1 Command

Tc = 0x58
Lc = 2 + 1
Vc = I || T

Parameters:

`I` := Object ID (2 bytes)

`T` := Type (1 byte)

11.3.2 Response

Tr = 0xd8
Lr = 0
Vr = Ø

CHAPTER

TWELVE

DERIVE ECDH COMMAND

Perform an ECDH operation.

12.1 Description

Perform an ECDH key exchange with the private key in the device.

12.2 Shell Example

Perform an ECDH operation with key `0x52b6` and a public key in the file `pubkey.pem`:

```
yubihsm> derive ecdh 0 0x52b6 pubkey.pem
   5898516bcb0cb3db89d53471137c2d1c741b8ba6ebf2bb01f4a62d97342e97b2
```

12.3 Protocol Details

12.3.1 Command

Tc = 0x57
Lc = 2 + LD
Vc = K || D

Parameters:

`I` := Object ID of the Asymmetric Key (2 bytes)

`D` := Uncompressed public key to perform the exchange with (57, 65, 97, 129 or 133 bytes)

12.3.2 Response

Tc = 0xd7
Lc = LX
Vc = X

Parameters:

X := X coordinate of the completed key exchange

CHAPTER

THIRTEEN

DEVICE INFO COMMAND

Get device metadata.

13.1 Description

Gets device version, device serial, supported Algorithms and available log entries.

13.2 Shell Example

Fetch device info for currently connected device:

```
yubihsm> get deviceinfo
Version number:          2.0.0
Serial number:           2000000
Log used:                2/62
Supported algorithms:    rsa-pkcs1-sha1, rsa-pkcs1-sha256, rsa-pkcs1-sha384,
                         rsa-pkcs1-sha512, rsa-pss-sha1, rsa-pss-sha256,
                         rsa-pss-sha384, rsa-pss-sha512, rsa2048,
                         rsa3072, rsa4096, ecp256, ecp384, ecp521, eck256,
                         ecbp256, ecbp384, ecbp512, hmac-sha1, hmac-sha256,
                         hmac-sha384, hmac-sha512, ecdsa-sha1, ecdh,
                         rsa-oaep-sha1, rsa-oaep-sha256, rsa-oaep-sha384,
                         rsa-oaep-sha512, aes128-ccm-wrap, opaque-data,
                         opaque-x509-certificate, mgf1-sha1, mgf1-sha256,
                         mgf1-sha384, mgf1-sha512, template-ssh,
                         aes128-yubico-otp, aes128-yubico-authentication,
                         aes192-yubico-otp, aes256-yubico-otp,
                         aes192-ccm-wrap, aes256-ccm-wrap,
                         ecdsa-sha256, ecdsa-sha384, ecdsa-sha512,
                         ed25519, ecp224,
```

13.3 Protocol Details

13.3.1 Command

Tc = 0x06
Lc = 0
Vc = Ø

13.3.2 Response

Tr = 0x86
Lr = 9 + algorithms
Vr = VMajor || VMinor || VBuild || S || Ltotal || Lused || A

Parameters:

`VMajor` := Major version number (1 byte)

`VMinor` := Minor version number (1 byte)

`VBuild` := Build version number (1 byte)

`S` := Serial number (4 bytes)

`Ltotal` := Log Store size expressed in number of log entries (1 byte)

`Lused` := Log lines used (1 byte)

`A` := List of supported Algorithms

CHAPTER

FOURTEEN

ECHO COMMAND

Echo data back from the device.

14.1 Description

Return the byte sequence present within the data field, without any modification. Can be sent over an encrypted Session or as a bare command.

14.2 Shell Example

Plain echo:

```
yubihsm> plain echo 0x3c 10
Response (10 bytes):
3c3c3c3c3c3c3c3c 3c3c
```

Echo over session 0:

```
yubihsm> echo 0 0x3c 10
Response (10 bytes):
3c3c3c3c3c3c3c3c 3c3c
```

14.3 Protocol Details

14.3.1 Command

Tc = 0x01
Lc = LE
Vc = E

Parameters:

E: Data to echo (1-2021 bytes)

14.3.2 Response

Tr = 0x81
Lr = LE
Vr = E

Parameters:

E: Data to echo (1-2021 bytes)

CHAPTER

FIFTEEN

EXPORT WRAPPED COMMAND

Get an Object in encrypted form.

15.1 Description

Retrieves an Object under wrap from the device. The Object is encrypted using AES-CCM with a 16 bytes MAC and a 13 bytes nonce.

15.2 Shell Example

Fetch the Asymmetric Key `0x997e` encrypted with Wrap Key `0xcf94` and store the result in the file `key.enc`:

```
yubihsm> get wrapped 0 0xcf94 asymmetric 0x997e key.enc
```

15.3 Protocol Details

15.3.1 Command

`Tc = 0x4a`
`Lc = 2 + 1 + 2`
`Vc = Iw || T || Io`

Parameters:

`Iw` := Object ID of Wrap Key to use (2 bytes)

`T` := Type of Object to wrap (1 byte)

`Io` := Object ID of Object to wrap (2 bytes)

15.3.2 Response

Tr = 0xca
Lr = 13 + LR
Vr = N || R

Parameters:

N := Nonce used for this wrap (13 bytes)

R := Wrapped data (Length dependent on object)

CHAPTER

SIXTEEN

GENERATE ASYMMETRIC KEY COMMAND

Generate an Asymmetric Key.

16.1 Description

Generate an Asymmetric Key in the device.

16.2 Shell Example

Generate a new key using `secp256r1` in the device:

```
yubihsm> generate asymmetric 0 0 eckey 1 sign-ecdsa ecp256
Generated Asymmetric key 0x2846
```

16.3 Protocol Details

16.3.1 Command

Tc = 0x46
Lc = 2 + 40 + 2 + 8 + 1
Vc = I || L || D || C || A

Generate an Asymmetric key-pair with a given ID. Each parameter has a fixed length and the order is compulsory.

Parameters:

`I` := Object ID of the Asymmetric Key (2 bytes)

`L` := Label (40 bytes)

`D` := Domains (2 bytes)

`C` := Capabilities (8 bytes)

`A` := Algorithm (1 byte)

16.3.2 Response

Tr = 0xc6
Lr = 2
Vr = I

Parameters:

I := Object ID of the created Asymmetric Key (2 bytes)

CHAPTER

SEVENTEEN

GENERATE HMAC KEY COMMAND

Generate an HMAC Key.

17.1 Description

Generate an HMAC Key in the device.

17.2 Shell Example

Generate an HMAC-SHA512 key:

```
yubihsm> generate hmackey 0 0 hmackey 1 sign-hmac:verify-hmac hmac-sha512
Generated HMAC key 0xa9bf
```

17.3 Protocol Details

17.3.1 Command

Tc = 0x5a
Lc = 2 + 40 + 2 + 8 + 1
Vr = I || L || D || C || A

Parameters:

I := Object ID of the HMAC Key (2 bytes)

L := Label (40 bytes)

D := Domains (2 bytes)

C := Capabilities (8 bytes)

A := Algorithm (1 byte)

17.3.2 Response

Tr = 0xda
Lr = 2
Vr = I

Parameters:

I:= Object ID of the created HMAC Key (2 bytes)

CHAPTER

EIGHTEEN

GENERATE OTP AEAD KEY COMMAND

Generate an OTP AEAD Key.

18.1 Description

Generate an OTP AEAD Key for Yubico OTP decryption.

18.2 Shell Example

Generate a new AES-256 OTP AEAD Key that can decrypt Yubico OTPs and create new AEADs:

```
yubihsm> generate otpaeadkey 0 0 otpaeadkey 1 decrypt-otp,
  create-otp-aead aes256-yubico-otp 0x01020304
Generated OTP AEAD key 0x027c
```

18.3 Protocol Details

18.3.1 Command

Tc = 0x66
Lc = 2 + 40 + 2 + 8 + 1 + 4
Vc = I || L || D || C || A || N

Parameters:

I := Object ID of the OTP AEAD Key (2 bytes)

L := Label (40 bytes)

D := Domains (2 bytes)

C := Capabilities (8 bytes)

A := Algorithm (1 byte)

N := Nonce ID (4 bytes)

18.3.2 Response

Tr = 0xe6
Lr = 2
Vr = I

Parameters:

I := Object ID of the created OTP AEAD Key (2 bytes)

CHAPTER

NINETEEN

GENERATE WRAP KEY COMMAND

Generate a Wrap Key.

19.1 Description

Generate a Wrap Key that can be used for export, import, wrap data and unwrap data.

19.2 Shell Example

Generate a new Wrap Key that can be used for `wrap` and `unwrap`:

```
yubihsm> generate wrapkey 0 0 wrapkey 1 wrap-data:unwrap-data none
  aes256-ccm-wrap
Generated Wrap key 0x5b3a
```

19.3 Protocol Details

19.3.1 Command

Tc = 0x5b
Lc = 2 + 40 + 2 + 8 + 1 + 8
Vc = I || L || D || C || A || DC

Parameters:

`I` := Object ID of the Wrap Key (2 bytes)

`L` := Label (40 bytes)

`D` := Domains (2 bytes)

`C` := Capabilities (8 bytes)

`A` := Algorithm (1 byte)

`DC` := Delegated Capabilities (8 bytes)

19.3.2 Response

Tr = 0xdb
Lr = 2
Vr = I

Parameters:

I := Object ID of created Wrap Key (2 bytes)

CHAPTER

TWENTY

GET LOG ENTRIES COMMAND

Fetch device audit log.

20.1 Description

Fetch all current entries from the device Log Store.

20.2 Shell Example

```
yubihsm> audit get 0
0 unlogged boots found
0 unlogged authentications found
Found 6 items
item:    46 -- cmd: 0x4b -- length:  234 -- session key: 0x0001
  -- target
key: 0xcf94 -- second key: 0x997e -- result: 0xcb -- tick: 335725
  -- hash: 415f51f1f035a1b713e730e4464e4033
item:    47 -- cmd: 0x4c -- length:   77 -- session key: 0x0001
  -- target
key: 0xaff7 -- second key: 0xffff -- result: 0xcc -- tick: 351714
  -- hash: 5496a60d478c2b9c801d8d32ca66b554
item:    48 -- cmd: 0x00 -- length:    0 -- session key: 0xffff
  -- target
key: 0x0000 -- second key: 0x0000 -- result: 0x00 -- tick: 0 -- hash:
   14ac7747ba9bbb243cfc70befeb5349b
item:    49 -- cmd: 0x03 -- length:   10 -- session key: 0xffff
  -- target
key: 0x0001 -- second key: 0xffff -- result: 0x83 -- tick: 139 -- hash:
   b20a8f25c025e693a8e869b433294a20
item:    50 -- cmd: 0x04 -- length:   17 -- session key: 0xffff
  -- target
key: 0x0001 -- second key: 0xffff -- result: 0x84 -- tick: 139 -- hash:
   ebfae425c319ac7a0afbb8b92597de7c
item:    51 -- cmd: 0x67 -- length:    2 -- session key: 0x0001
  -- target
key: 0xffff -- second key: 0xffff -- result: 0xe7 -- tick: 697 -- hash:
   2e395d1b706668737e1d2215813db47e
```

20.3 Protocol Details

20.3.1 Command

Tc = 0x4d
Lc = 0
Vc = Ø

20.3.2 Response

Tr = 0xcd
Lr = 2 + 2 + 1 + (N * 32)
Vr = B || O || N || E1 || E2 || ... || EN

Parameters:

`B` := Number of unlogged boot events (if the log buffer is full and audit enforce is set) (2 bytes)

`O` := Number of unlogged authentication events (if the log buffer is full and audit enforce is set) (2 bytes)

`N` := Number of elements in the list (1 byte)

`Ei`:= Generic log entry composed of

- Command number (two bytes)
- Command ID (one byte)
- Command length (two bytes)
- ID of the originating session's authentication key (two bytes)
- Target key affected by the command (two bytes)
- Secondary key if the command affected more than one key (two bytes)
- Result of the command on success or an error code if unsuccessful (one byte)
- Systick when the command was processed (4 bytes)
- Digest (16 bytes)

The digest is computed as `trunc(16, SHA256(Ei.Data || trunc(16, Ei-1.Digest)))`. For the initial log entry, a random string of 32 bytes is used, instead of the digest of the previous message.

When the device initializes after a reset, a log entry with all fields set to `0xff` is logged.

When the device boots up, a log entry with all fields set to `0x00` is logged.

CHAPTER

TWENTYONE

GET OBJECT INFO COMMAND

Get Object metadata.

21.1 Description

Fetch all metadata about an Object

21.2 Shell Example

Get Object info for Asymmetric Key with ID `0x1e15`:

```
yubihsm> get objectinfo 0 0x1e15 asymmetric-key
id: 0x1e15, type: asymmetric-key, algorithm: rsa2048, label: "rsakey",
  length: 896, domains: 1, sequence: 0, origin: imported, capabilities:
  sign-pkcs
```

21.3 Protocol Details

21.3.1 Command

Tc = 0x4e
Lc = 2 + 1
Vc = I || T

Parameters:

I := Object ID (2 bytes)

T := Type (1 byte)

21.3.2 Response

Tr = 0xce
Lr = 8 + 2 + 2 + 2 + 1 + 1 + 1 + 1 + 40 + 8
Vr = C || I || N || D || T || A || S || O || L || DC

Parameters:

C := Capabilities (8 bytes)

I := Object ID (2 bytes)

N := Object Length (2 bytes)

D := Domains (2 bytes)

T := Type (1 byte)

A := Algorithm (1 byte)

S := Sequence (1 byte)

O := Origin (1 byte)

L := Label (40 bytes)

DC := Delegated Capabilities (8 bytes)

CHAPTER

TWENTYTWO

GET OPAQUE COMMAND

Fetch an Opaque Object from device.

22.1 Description

Retrieve an Opaque Object (like an X.509 certificate) from the device.

22.2 Shell Example

Fetch Opaque Object `0xe255` and store in the file `cert.der`:

```
yubihsm> get opaque 0 0xe255 cert.der
```

22.3 Protocol Details

22.3.1 Command

Tc = 0x43
Lc = 2
Vc = I

Parameters:

`I` := Object ID (2 bytes)

22.3.2 Response

Tr = 0xc3
Lr = LD
Vr = D

Parameters:

`D` := Data

CHAPTER

TWENTYTHREE

GET OPTION COMMAND

Fetch a device-global option.

23.1 Description

Get device-global Options. Each invocation of this command retrieves a single Option, which is selected by its represented TAG (see *SET OPTION Command*).

23.2 Shell Example

```
yubihsm> get option 0 force-audit
Option value is: 00
```

23.3 Protocol Details

23.3.1 Command

Tc = 0x50
Lc = 1
Vc = T

Parameters:

T := The tag of the selected option (1 byte)

23.3.2 Response

Tr = 0xd0
Lr = LO
Vr = O

Parameters:

O := The option-specific value (LO bytes)

CHAPTER

TWENTYFOUR

GET PSEUDO RANDOM COMMAND

Get pseudo-random data from device.

24.1 Description

Extract a fixed number of pseudo-random bytes from the device, using the internal PRNG.

24.2 Shell Example

```
yubihsm> get random 0 16
bd50979da2d1bca13d8d735abf419556
```

24.3 Protocol Details

24.3.1 Command

Tc = 0x51
Lc = 2
Vc = B

Parameters:

B := Number of pseudo-random bytes to extract (2 bytes)

24.3.2 Response

Tr = 0xd1
Lr = B
Vr = R

Parameters:

R := Random data (B bytes)

CHAPTER

TWENTYFIVE

GET PUBLIC KEY COMMAND

Fetch a public key from device.

25.1 Description

Fetch the public key of an Asymmetric Key.

25.2 Shell Example

Fetch the public key of Asymmetric Key `0x2846`:

```
yubihsm> get pubkey 0 0x2846
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE85fayPHTvCrv0RRcyCsHv0hTKAM7
xHiU2I3NgO61RTRQumGDeBnQZIITykK/0PWKLGDANfBVrmKkWWxB47ze9A==
-----END PUBLIC KEY-----
```

25.3 Protocol Details

25.3.1 Command

Tc = 0x54
Lc = 2
Vc = I

Parameters:

I := Object ID of the Asymmetric Key (2 bytes)

25.3.2 Response

Tr = 0xd4
Lr = 1 + LP1 { + LP2 }
Vr = A || P1 { || P2 }

Parameters:

`A` := Algorithm (1 byte)

`P1` :=

For RSA: Public modulus N (256, 384 or 512 bytes)

For ECC: Public point X (32, 48, 64 or 66 bytes)

For EDC: Public point A, compressed (32 bytes)

`P2` :=

For RSA: NOT DEFINED

For ECC: Public point Y (32, 48, 64 or 66 bytes)

For EDC: NOT DEFINED

CHAPTER

TWENTYSIX

GET STORAGE INFO COMMAND

Fetch storage information.

26.1 Description

Report currently free storage. This is reported as currently free records, free pages and page size. Each object takes a record slot and will use as many pages as needed.

26.2 Shell Example

```
yubihsm> get storage 0
free records: 255/256, free pages: 1023/1024 page size: 126 bytes
```

26.3 Protocol Details

26.3.1 Command

Tc = 0x41
Lc = 0
Vc = Ø

26.3.2 Response

Tr = 0xc1
Lr = 10
Vr = Rtotal || Rfree || Ptotal || Pfree || S

Parameters:

`Rtotal` := Total number of records (2 bytes)

`Rfree` := Currently free storage records (2 bytes)

`Ptotal` := Total number of pages (2 bytes)

`Pfree` := Currently free storage pages (2 bytes)

`S` := Page size in bytes (2 bytes)

CHAPTER

TWENTYSEVEN

GET TEMPLATE COMMAND

Fetch a Template Object from the device.

27.1 Description

Retrieve a Template Object from the device.

27.2 Shell Example

Fetch Template Object `0x7b19` and store in the file `template.dat`:

```
yubihsm> get template 0 0x7b19 template.dat
```

27.3 Protocol Details

27.3.1 Command

Tc = 0x5f
Lc = 2
Vc = I

Parameters:

`I` := Object ID of the Template to retrieve (2 bytes)

27.3.2 Response

Tr = 0xdf
Lr = LD
Vr = D

Parameters:

`D` := Data

CHAPTER

TWENTYEIGHT

IMPORT WRAPPED COMMAND

Import an wrapped/encrypted object into the device.

28.1 Description

Import a wrapped/encrypted Object that was previously exported by an YubiHSM 2 device. The imported object will retain its metadata (Object ID, Domains, Capabilities …etc), however, the object's origin will be marked as *imported* instead of *generated*.

28.2 Shell Example

Import the Object stored in `key.enc` and unwrap it using Wrap Key `0xcf94`

```
yubihsm> put wrapped 0 0xcf94 key.enc
Object imported as 0x997e of type asymmetric
```

28.3 Protocol Details

28.3.1 Command

Tc = 0x4b
Lc = 2 + 13 + LO
Vc = I || N || O

Parameters:

`I` := Object ID of the Wrap Key (2 bytes)

`N` := Nonce associated with this wrapped Object (13 bytes)

`O` := Wrapped Object (Length dependant on Object)

28.3.2 Response

Tc = 0xcb
Lc = 3
Vc = T || I

Parameters:

T := Type of imported Object (1 byte)

I := Object ID of imported Object (2 bytes)

CHAPTER

TWENTYNINE

LIST OBJECTS COMMAND

List Objects in device.

29.1 Description

Get a filtered list of Objects from the device.

29.2 Shell Example

Get a list of all Asymmetric Keys for Session 0:

```
yubihsm> list objects 0 0 asymmetric-key
Found 4 object(s)
id: 0x3479, type: asymmetric-key, sequence: 0
id: 0x7df6, type: asymmetric-key, sequence: 0
id: 0x9602, type: asymmetric-key, sequence: 0
id: 0xd6cd, type: asymmetric-key, sequence: 0
```

29.3 Protocol Details

29.3.1 Command

Tc = 0x48
Lc = LF
Vc = F

Parameters:

F := List of Tag-Value pairs describing a filter to apply. Possible tags to use for filtering are described in the table below.

Name	Identifier	Length
ID	0x01	2 bytes
TYPE	0x02	1 byte
DOMAINS	0x03	2 bytes
CAPABILITIES	0x04	8 bytes
ALGORITHM	0x05	1 byte
LABEL	0x06	40 bytes

29.3.2 Response

`Tr = 0xc8`
`Lr = 4 * N`
`Vr = R1 || R2 || ... || RN`

Parameters:

`Ri` := Object ID (2 bytes), Type (1 byte) and Sequence (1 byte).

CHAPTER

THIRTY

PUT ASYMMETRIC KEY COMMAND

Import an Asymmetric Key.

30.1 Description

Import an Asymmetric Key into the device.

30.2 Shell Example

Store an RSA key from `key.pem` into the device:

```
yubihsm> put asymmetric 0 0 rsakey 1 sign-pkcs key.pem
Stored Asymmetric key 0x1e15
```

30.3 Protocol Details

30.3.1 Command

Tc = 0x45
Lc = 2 + 40 + 2 + 8 + 1 + LP1 { + LP2 }
Vc = I || L || D || C || A || P1 { || P2 }

The key parameters vary according to the chosen algorithm. Each parameter has a fixed length and the order is compulsory.

Parameters:

`I` := Object ID of the Asymmetric Key (2 bytes)

`L` := Label (40 bytes)

`D` := Domains (2 bytes)

`C` := Capabilities (8 bytes)

`A` := Algorithm (1 byte)

`P1` :=

For RSA: secret prime p (128, 192 or 256 bytes)

For ECC: private key integer d (32, 48, 64 or 66 bytes)

For EDC: private key integer k (32 bytes)

`P2` :=

For RSA: secret prime q (128, 192 or 256 bytes)

For ECC: NOT DEFINED

For EDC: NOT DEFINED

30.3.2 Response

`Tr = 0xc5`
`Lr = 2`
`Vr = I`

Parameters:

`I` := ID of created Object (2 bytes)

CHAPTER

THIRTYONE

PUT AUTHENTICATION KEY COMMAND

Store a new Authentication Key.

31.1 Description

Store an Authentication Key in the device.

31.2 Shell Example

Store a new Authentication Key derived from the password `newpassword`:

```
yubihsm> put authkey 0 0 authkey 1 generate-asymmetric-key,sign-pkcs
   sign-pkcs newpassword
Stored Authentication key 0xbb72
```

31.3 Protocol Details

31.3.1 Command

Tc = 0x44
Lc = 2 + 40 + 2 + 8 + 1 + 8 + 16 + 16
Vc = I || L || D || C || A || DC || Ke || Km

Parameters:

`I` := Object ID of the Authentication Key (2 bytes)

`L` := Label (40 bytes)

`D` := Domains (2 bytes)

`C` := Capabilities (8 bytes)

`A` := Algorithm (1 byte)

`DC` := Delegated Capabilities (8 bytes)

`Ke` := Encryption Key (16 bytes)

`Km` := Mac Key (16 bytes)

31.3.2 Response

Tr = 0xc4
Lr = 2
Vr = I

Parameters:

I := Object ID of created Authentication Key (2 bytes)

CHAPTER

THIRTYTWO

PUT HMAC KEY COMMAND

Import an HMAC Key.

32.1 Description

Store an HMAC Key in the device.

32.2 Shell Example

Store an HMAC Key with the binary value `666f6f` in the device:

```
yubihsm> put hmackey 0 0 hmackey 1 sign-hmac,verify-hmac hmac-sha256 666f6f
Stored HMAC key 0x7cf2
```

32.3 Protocol Details

32.3.1 Command

Tc = 0x52
Lc = 2 + 40 + 2 + 8 + 1 + LP
Vc = I || L || D || C || A || P

Parameters:

I := Object ID of the HMAC Key (2 bytes)

L := Label (40 bytes)

D := Domains (2 bytes)

C := Capabilities (8 bytes)

A := Algorithm (1 byte)

P := Key (Minimum 1 byte)

For HMAC-SHA1 and HMAC-SHA256: maximum 64 bytes

For HMAC-SHA384 and HMAC-SHA512: maximum 128 bytes

32.3.2 Response

Tr = 0xd2
Lr = 2
Vr = I

Parameters:

I := Object ID of created HMAC Key (2 bytes)

CHAPTER

THIRTYTHREE

PUT OPAQUE COMMAND

Store an Opaque Object.

33.1 Description

Stores Opaque data (like an X.509 certificate) in the device. The size of the object is currently limited to what will fit into one message to the YubiHSM 2 (2028 bytes, including the headers).

33.2 Shell Example

Store the certificate in file `cert.der` in the device:

```
yubihsm> put opaque 0 0 certificate 1 none opaque-x509-certificate cert.der
Stored Opaque object 0xe255
```

33.3 Protocol Details

33.3.1 Command

Tc = 0x42
Lc = 2 + 40 + 2 + 8 + 1 + LO
Vc = I || L || D || C || A || O

Parameters:

I := Object ID (2 bytes)

L := Label (40 bytes)

D := Domains (2 bytes)

C := Capabilities (8 bytes)

A := Algorithm (1 byte)

O := Opaque data

33.3.2 Response

Tr = 0xc2
Lr = 2
Vr = I

Parameters:

I := Object ID of created Opaque Object (2 bytes)

CHAPTER

THIRTYFOUR

PUT OTP AEAD KEY COMMAND

Import an OTP AEAD Key.

34.1 Description

Import an OTP AEAD Key used for Yubico OTP Decryption.

34.2 Shell Example

Import OTP AEAD Key with Nonce ID `0x01020304` and key value `000102030405060708090a0b0c0d0e0f` (AES-128):

```
yubihsm> put otpaeadkey 0 0 otpaeadkey 1 decrypt-otp 0x01020304␣
↪000102030405060708090a0b0c0d0e0f
Stored OTP AEAD key 0xe34f
```

34.3 Protocol Details

34.3.1 Command

Tc = 0x65
Lc = 2 + 40 + 2 + 8 + 1 + 4 + LK
Vc = I || L || D || C || A || N || K

Parameters:

I := Object ID (2 bytes)

L := Label (40 bytes)

D := Domains (2 bytes)

C := Capabilities (8 bytes)

A := Algorithm (1 byte)

N := Nonce ID (4 bytes)

K := Key (16, 24 or 32 bytes depending on algorithm)

34.3.2 Response

Tr = 0xe5
Lr = 2
Vr = I

Parameters:

I := ID of created OTP AEAD Key (2 bytes)

CHAPTER

THIRTYFIVE

PUT TEMPLATE COMMAND

Store a Template.

35.1 Description

Stores a Template in the device. The size of the object is currently limited to what will fit into one message to the YubiHSM (2021 bytes, including the headers).

35.2 Shell Example

Store the SSH Template in file `template.dat` in the device:

```
yubihsm> put template 0 0 ssh_template 1 none template-ssh template.dat
Stored Template object 0x7b19
```

35.3 Protocol Details

35.3.1 Command

Tc = 0x5e
Lc = 2 + 40 + 2 + 8 + 1 + LD
Vc = I || L || D || C || A || D

Parameters:

I := Object ID of the Template (2 bytes)

L := Label (40 bytes)

D := Domains (2 bytes)

C := Capabilities (8 bytes)

A := Algorithm (1 byte)

D := Template data

35.3.2 Response

Tr = 0xde
Lr = 2
Vr = I

Parameters:

I := Object ID of created Template (2 bytes)

CHAPTER

THIRTYSIX

PUT WRAP KEY COMMAND

Import a Wrap Key.

36.1 Description

Import a key for wrapping into the device.

36.2 Shell Example

Import an AES-128 Wrap Key able to export and import, with some Delegated Capabilities set:

```
yubihsm> put wrapkey 0 0 wrapkey 1 export-wrapped,import-wrapped
   exportable-under-wrap,sign-pkcs,sign-pss 000102030405060708090a0b0c0d0e0f
Stored Wrap key 0xaff7
```

36.3 Protocol Details

36.3.1 Command

Tc = 0x4c
Lc = 2 + 40 + 2 + 8 + 1 + 8 + LW
Vc = I || L || D || C || A || DC || W

Parameters:

I := Object ID (2 bytes)

L := Label (40 bytes)

D := Domains (2 bytes)

C := Capabilities (8 bytes)

A := Algorithm (1 byte)

DC := Delegated Capabilities (8 bytes)

W := Wrap Key (16, 24 or 32 bytes)

For AES128_CCM_WRAP: 16 bytes

For AES192_CCM_WRAP: 24 bytes

For AES256_CCM_WRAP: 32 bytes

36.3.2 Response

Tc = 0xcc
Lc = 2
Vc = I

Parameters:

I := ID of created Wrap Key (2 bytes)

CHAPTER

THIRTYSEVEN

RANDOMIZE OTP AEAD COMMAND

Create an OTP AEAD from random data.

37.1 Description

Create a new OTP AEAD using random data for key and private ID.

37.2 Shell Example

Generate a new OTP AEAD using OTP AEAD Key `0xc5f4` and put the result in file aead:

```
yubihsm> otp aead_random 0 0xc5f4 aead
```

37.3 Protocol Details

37.3.1 Command

Tc = 0x62
Lc = 2
Vc = I

Parameters:

`I` := Object ID for the OTP AEAD Key (2 bytes)

37.3.2 Response

Tr = 0xe2
Lr = 36
Vr = A

Parameters:

`A` := Nonce concatenated with AEAD (36 bytes)

CHAPTER

THIRTYEIGHT

RESET DEVICE COMMAND

Factory reset a device.

38.1 Description

Resets and reboots the device, deletes all Objects and restores the default Options and Authentication Key.

38.2 Shell Example

Send reset over Session `0`:

```
yubihsm> reset 0
Device successfully reset
```

38.3 Protocol Details

38.3.1 Command

Tc = 0x08
Lc = 0
Vc = Ø

38.3.2 Response

Tr = 0x88
Lr = 0
Vr = Ø

CHAPTER

THIRTYNINE

REWRAP OTP AEAD COMMAND

Rewrap an OTP AEAD.

39.1 Description

Re-encrypt a Yubico OTP AEAD from one OTP AEAD Key to another OTP AEAD Key.

39.2 Shell Example

N/A

39.3 Protocol Details

39.3.1 Command

`Tc = 0x63`
`Lc = 2 + 2 + 36`
`Vc = I1 || I2 || A`

Parameters:

`I1` := Key ID from (2 bytes)

`I2` := Key ID to (2 bytes)

`A` := Nonce concatenated with AEAD (36 bytes)

39.3.2 Response

Tr = 0xe3
Lr = 36
Vr = A

Parameters:

`A` := Nonce concatenated with AEAD (36 bytes)

CHAPTER

FORTY

SESSION MESSAGE COMMAND

Send a command over an established session.

40.1 Description

Sends a wrapped command for a previously established session. The command is encrypted and authenticated.

40.2 Shell Example

Send an echo over Session `0`:

```
yubihsm> echo 0 0xff 1
Response (1 bytes):
ff
```

40.3 Protocol Details

40.3.1 Command

`Tc = 0x05`
`Lc = 1 + Linner_c + 8`
`Vc = S || Ic || Mc`

Parameters:

`S` := Session ID (1 byte)

`Linner_c/inner_r` := Length of the encrypted inner command / response (2 bytes)

`Mc/r` := CMAC of the outer command / response (8 bytes)

40.3.2 Response

Tr = 0x85
Lr = 1 + Linner_r + 8
Vr = S || Ir || Mr

CHAPTER

FORTYONE

SET LOG INDEX COMMAND

Set the last extracted log entry.

41.1 Description

Inform the device what the last extracted log entry is so logs can be reused. Mostly of practical use when forced auditing is enabled.

41.2 Shell Example

Set log index `41` as the last extracted entry:

```
yubihsm> audit set 0 41
```

41.3 Protocol Details

41.3.1 Command

Tc = 0x67
Lc = 2
Vc = I

Parameters:

`I` := Index to set as last read log (2 bytes)

41.3.2 Response

Tr = 0xe7
Lr = 0
Vr = Ø

CHAPTER

FORTYTWO

SET OPTION COMMAND

Set a device-global option.

42.1 Description

Set device-global options that affect general behavior. Each invocation of this command sets a single option, which is represented as a `TAG-LENGTH-VALUE (TLV)`.

42.2 Shell Example

Turn off audit logging for Sign HMAC (command `53`) and Verify HMAC (command `5c`):

```
yubihsm> put option 0 command_audit 53005c00
```

42.3 Protocol Details

42.3.1 Command

Tc = 0x4f
Lc = 3 + Lo
Vc = TO

Parameters:

`To` := The TLV encoding of the selected option

`Lo` := The option-specific length in bytes

The options currently supported are the following:

`TAG` is 1 byte

`LENGTH` is 2 bytes

`VALUE` is Lo bytes

Tags:

Force audit	0x01
Command audit	0x03

Values:

OFF	0x00	Disabled
ON	0x01	Enabled
FIX	0x02	Enabled, only possible to turn off through factory reset

There are two supported options, `Force audit` and `Command audit`.

With `Force audit` set, the device will refuse operations as long as the Log Store is full. It takes a 1 byte value option.

`Command audit` can be used to toggle whether a specific command should be logged, this takes tuples of command number and option value.

42.3.2 Response

`Tr = 0xcf`
`Lr = 0`
`Vr = Ø`

CHAPTER

FORTYTHREE

SIGN ATTESTATION CERTIFICATE COMMAND

Attest an Asymmetric Key.

43.1 Description

Get attestation of an Asymmetric Key, output is an X.509 certificate.

43.2 Shell Example

Attest Asymmetric Key `0x79c3` using attestation key `0` (builtin):

```
yubihsm> attest asymmetric 0 0x79c3 0
-----BEGIN CERTIFICATE-----
MIIDeTCCAmGgAwIBAgIQaa8FkvRhqntp5HjyyCfilzANBgkqhkiG9w0BAQsFADAn
MSUwIwYDVQQDDBxZdWJpSFNNIEF0dGVzdGF0aW9uICgxMjM0NTYpMCAXDTE3MDEw
MTAwMDAwMFoYDzIwNzExMDA1MDAwMDAwWjAoMSYwJAYDVQQDDB1ZdWJpSFNNIEF0
dGVzdGF0aW9uIGlkOjB4NzljMzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
ggEBAMYpAzHar0syanQEiRqWy8WDO5qETjDulo2txNBDwyMCNgeEYzo/uglUXLEm
Zj6Dd8EcdY9upHoqVpLduB+GIt+UEq5DeMN5Rzj2QZ/lQMELMdaD9ODc707aPvKT
/oAuj1aZ89vfg7jEVWBTPWquyFaxaCBoz8WWta9j5JxRppQpR27ub43950fX3wpW
btvlNLMx0QAQdDqEm2V3TEhnbo6T5XsgC78OdOikyJw2TP062rQXSY7GRuXob/Qa
INsJRXbbydqUXDHFNq8GnSkL8dHsNdf7bOSdAV6Vlw30JFbJ2uoW2EkGmF9qYWnt
EVyyPMMQwF09r9HVpLF83TBaYoMCAwEAAaOBnTCBmjATBgorBgEEAYLECgQBBAUE
AwIAADATBgorBgEEAYLECgQCBAUCAx6EgDASBgorBgEEAYLECgQDBAQDAgABMBMG
CisGAQQBgsQKBAQEBQMDAAABMBkGCisGAQQBgsQKBAUECwMJAAAAAAAAAAARAMBIG
CisGAQQBgsQKBAYEBAICecMwFgYKKwYBBAGCxAoECQQIDAZyc2FrZXkwDQYJKoZI
hvcNAQELBQADggEBABRReYze+KRfevrgyI3C2aLAWSiQRjJ6vvaP1Fh4bOw4X2HC
rLAI150h4O5eH/aXVNv+368FWlQhcY68jKDgDoeckrlt9thFxaphasd/Wt1Pbqzj
trnEillYjjP6rddyCR1yitmnQ3Qnsk3w1mTE/AtzmDOi7V/wNymilB79OFDGmB6P
d1VI7zGUHtLlj1qeyY4/ETqKuPDzZY5RUPYrO8/iPzy64AdtDXt1e39n9pTcohp2
PSQQe36gU7vt9+5SebEj0CF/qTk317L1R42TfeHFSJlgBTHSWcuvDORNJxDHTcco
bI+wE2dCcnjyLU9dr5tkNsD3k5pscuTmpBGFDlg=
-----END CERTIFICATE-----
```

43.3 Protocol Details

43.3.1 Command

Tc = 0x64
Lc = 2 + 2
Vc = I || A

Parameters:

I := Object ID of the Asymmetric Key to attest (2 bytes)

A := Object ID of the Asymmetric Key used for attestation (2 bytes)

If A is 0 the internal attestation key is used.

43.3.2 Response

Tr = 0xe4
Lr = LX
Vr = X

Parameters:

X := DER encoded X.509 attestation

CHAPTER

FORTYFOUR

SIGN ECDSA COMMAND

Sign data with ECDSA.

44.1 Description

Computes a digital signature using ECDSA on the provided data.

44.2 Shell Example

Sign data in file `data` using key `0x52b6` and put the result in file `sig`:

```
yubihsm> sign ecdsa 0 0x52b6 ecdsa-sha256 data sig
```

44.3 Protocol Details

44.3.1 Command

Tc = 0x56
Lc = 2 + LD
Vc = I || D

Parameters:

`I` := Object ID of the Asymmetric Key (2 bytes)

`D` := H

The DSI for ECDSA is a possibly zero-left-padded hash of the data, `H`.

44.3.2 Response

Tr = 0xd6
Lr = LDS
Vr = DS

Parameters:

DS := Resulting signature

The length of DS, LDS, depends on the Algorithm used and equals the length of the signature plus its DER encoding.

CHAPTER

FORTYFIVE

SIGN EDDSA COMMAND

Sign with EdDSA.

45.1 Description

Computes a digital signature using EdDSA on the provided data.

45.2 Shell Example

Perform an EdDSA signature with key `0xddf6` of the content of file `data`:

```
yubihsm> sign eddsa 0 0xddf6 ed25519 data
   wZljrOstOLPuMHGrXDnpAb5Wxo79+wX/vQkb/6K34tOd8se/QfLNRVTonfErttkWUAz/
↪UlNtaG4XJYnY8vabCQ==
```

45.3 Protocol Details

45.3.1 Command

Tc = 0x6a
Lc = 2 + LD
Vc = I || D

Parameters:

`I` := Object ID of the Asymmetric Key (2 bytes)

The DSI for EdDSA is the raw data D.

`DSI` := D

For a given DSI, the command will generate a digital signature `DS`. The length of `DS`, `LDS`, depends on the Algorithm used. At this time only Ed25519 is implemented.

`DS` := EdDSA(DSI). Key is omitted

`DS` := 0x0040 bytes

45.3.2 Response

Tr = 0xea
Lr = LDS
Vr = DS

Parameters:

DS := Resulting signature

CHAPTER

FORTYSIX

SIGN HMAC COMMAND

HMAC data.

46.1 Description

Perform an HMAC operation in device and return the result.

46.2 Shell Example

Perform an HMAC operation using the HMAC Key `0x7cf2`:

```
yubihsm> hmac 0 0x7cf2 666f6f626172␣
↪4c17e17300a51a3f8aeeba131e9c680e4e40b429aa1d547807efd8e3d95ccd39
```

46.3 Protocol Details

46.3.1 Command

Tc = 0x53
Lc = 2 + LD
Vc = I || D

Parameters:

`I` := Object ID of the HMAC Key (2 bytes)

`D` := Data to HMAC

46.3.2 Response

Tr = 0xd3
Lr = LR
Vr = R

Parameters:

R := HMAC Response, 20, 32, 48 or 64 bytes depending on the Algorithm.

CHAPTER

FORTYSEVEN

SIGN PKCS1 COMMAND

Sign with RSA-PKCS#1v1.5.

47.1 Description

Computes a digital signature using RSA-PKCS1v1.5 on the provided data.

47.2 Shell Example

Sign the data in the file `test` using rsa-pkcs1-sha256:

```
yubihsm> sign pkcs1v1_5 0 0x1e15 rsa-pkcs1-sha256 test
   eu9HQceSs0zsUogVloovRRcDGtkBj5AIp2Nnk6LWT4KbQZX8ac+vmFtVotjDIF9PkQ9MA8K
   sfUGvXAxpnvUyin3BjGvzENu5XRi+ZOGP4m8777zbDi1v7FKQSx8/KdZf4tulIsL4rM4M+uH
   /QoQ83vWty4c63QjcSlZJQDsdHn9r3E5or3QgBo06yK2Rd8W3WYGloSPvDaGu7L87CDFy
   MniAQB//Sw7bYr4hbVpKIWi6q4VPhBKdaB6+FzTmYrqsSv1vwek0V4LbvyelTHlh9PpFuSF
   ZeGJ/i1gkIeSO2XlKNLa4+AO+H+TYUOP3b6Qlhs3f7e4AFFWKE6lPpDHJA==
```

47.3 Protocol Details

47.3.1 Command

Tc = 0x047
Lc = 2 + LD
Vc = I || D

Parameters:

`I` := Object ID of the Asymmetric Key (2 bytes)

`D` := Digest

The Digest can be either a raw hash of data, where DigestInfo will be applied in the device, or DigestInfo + hash. Hashes supported are SHA-1, SHA-256, SHA-384 and SHA-512.

47.3.2 Response

Tr = 0xc7
Lr = LDS
Vr = DS

Parameters:

DS := Resulting signature

CHAPTER

FORTYEIGHT

SIGN PSS

Sign data using RSA-PSS.

48.1 Description

Computes a digital signature using RSA-PSS on the provided data.

48.2 Shell Example

Sign what is in file `data` using key `0x79c3` and put the resulting signature in `sig`:

```
yubihsm> sign pss 0 0x79c3 rsa-pss-sha256 data sig
```

48.3 Protocol Details

48.3.1 Command

Tc = 0x55
Lc = 2 + 1 + 2 + LD
Vc = I || M || S || D

Parameters:

`I` := Object ID of the Asymmetric Key (2 bytes)

`M` := Hash Algorithm to use for MGF1

`S` := Salt len (2 bytes)

`D` := Hashed data (20, 32, 48 or 64 bytes)

The DSI of EMSA-PSS is as defined in RFC 3447:

`DSI` := EMSA-PSS-ENCODE(M, emBits, Hash, MGF, sLen).

`Hash` is a supported hash Algorithm

`MGF` is a supported masking function

`sLen` is the length of the Salt

The DSI is generated internally and only the Hash of the data and the Salt length are provided.

48.3.2 Response

Tr = 0xd5
Lr = LDS
Vr = DS

Parameters:

DS := Resulting signature

CHAPTER

FORTYNINE

SIGN SSH CERTIFICATE COMMAND

Sign an SSH Certificate request.

49.1 Description

Produce an SSH Certificate signature. The certificate can then be used to login to hosts.

49.2 Shell Example

Produce a new SSH Certificate.

```
yubihsm> certify 0 0xabcd 0x1234 rsa-pkcs-sha256 req.dat cert.dat
```

49.3 Protocol Details

49.3.1 Command

Tc = 0x5d
Lc = 2 + 2 + 1 + 4 + 256 + LR
Vc = I || T || A || N || S || R

Sign and SSH Certificate by using the given Asymmetric Key and SSH Template.

Parameters:

I := Object ID of the Asymmetric Key (2 bytes)

T := Object ID of the SSH Template (2 bytes)

A := Algorithm (1 byte)

N := Timestamp with the definition of Now (4 bytes)

S := Signature over the request and timestamp (256 bytes)

R := Request (LR bytes)

49.3.2 Response

Tr = 0xd6
Lr = LS
Vr = S

Parameters:

S := Certificate Signature (LS bytes)

CHAPTER

FIFTY

UNWRAP DATA COMMAND

Decrypt data.

50.1 Description

Decrypt (unwrap) data using a Wrap Key.

50.2 Shell Example

```
yubihsm> decrypt aesccm 0 0x5b3a MRkj6B0AAAAAAAAAAoO4dkIeAYoPvwTV/M/JX1dwKnLqnERO1hSW4wPS
Hello world!
```

50.3 Protocol Details

50.3.1 Command

Tc = 0x69
Lc = 2 + 13 + LD + 16
Vc = I || N || D || M

Parameters:

I := Object ID of a Wrap Key (2 bytes)

N := Nonce (13 bytes)

D := Data to be unwrapped

M := Mac (16 bytes)

50.3.2 Response

Tr = 0xe9
Lr = LD
Vr = D

Parameters:

D := Unwrapped data

CHAPTER

FIFTYONE

VERIFY HMAC COMMAND

Verify an HMAC.

51.1 Description

Verify a generated HMAC.

51.2 Shell Example

N/A

51.3 Protocol Details

51.3.1 Command

Tc = 0x5c
Lc = 2 + LH + LD
Vc = I || H || D

Parameters:

I := Object ID of the HMAC Key (2 bytes)

H := HMAC (20, 32, 48 or 64 bytes)

D := Data

51.3.2 Response

Tr = 0xdc
Lr = 1
Vr = V

Parameters:

V := Verified (1 byte)

V will have the value 1 if verification succeeded and 0 otherwise.

CHAPTER

FIFTYTWO

WRAP DATA COMMAND

Encrypt data.

52.1 Description

Encrypt (wrap) data using a Wrap Key.

52.2 Shell Example

Using Wrap Key `0x5b3a` encrypt the string "Hello world!":

```
yubihsm> encrypt aesccm 0 0x5b3a "Hello world!"
MRkj6B0AAAAAAAAAAoO4dkIeAYoPvwTV/M/JX1dwKnLqnERO1hSW4wPS
```

52.3 Protocol Details

52.3.1 Command

Tc = 0x68
Lc = 2 + LD
Vc = I || D

Parameters:

`I` := Object ID of the Wrap Key (2 bytes)

`D` := Data to be wrapped

52.3.2 Response

Tr = 0xe8
Lr = 13 + LD + 16
Vr = N || D || M

Parameters:

N := Nonce (13 bytes)

D := Wrapped data

M := Mac (16 bytes)

CHAPTER

FIFTYTHREE

COPYRIGHT

Trademarks

Yubico and YubiKey are registered trademarks of Yubico AB. All other trademarks are the property of their respective owners.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design, and manufacturing. Yubico shall have no liability for any error or damages of any kind resulting from the use of this document.

The Yubico Software referenced in this document is licensed to you under the terms and conditions accompanying the software or as otherwise agreed between you or the company that you are representing.

Contact Information

Yubico Inc.
530 Lytton Street
Suite 301
Palo Alto, CA 94301
USA

Click the links to:

- Submit a support request
- Send a Contact Me request
- See additional contact options for getting touch with us

Document Updated

2022-05-11 23:17:23 UTC