TRUSTED® COMPUTING GROUP

REFERENCE

TCG Storage Application Note: Encrypting Drives Compliant with Key Per I/O SSC

Version 1.00
Revision 1.11
October 23, 2023

Contact: admin@trustedcomputinggroup.org

PUBLISHED

DISCLAIMERS, NOTICES, AND LICENSE TERMS

ACKNOWLEDGEMENT

The TCG wishes to thank all those who contributed to this specification. This document builds on work done in various working groups in the TCG and the industry at large.

Name	Company
Chandra Nelogal	Dell Technologies, Inc.
Paul Suhler	Kioxia Corporation
Taku Kato	Kioxia Corporation
Alan Arnold	Lenovo Inc.
Michael McDonnell	Marvell Technology
Alon Cohen	Microchip Technology
Walt Hubis	Micron Technology, Inc.
Fred Knight	NetApp
Sridhar Balasubramanian	NetApp
Eric Hibbard	Samsung Semiconductor Inc.
Anthony Duran	Seagate Technology
Jim Hatfield	Seagate Technology
Festus Hategekimana	Solidigm
John Mathews	Solidigm
Santosh Kumar	Solidigm
Patrick Hery	Toshiba Corporation
Joseph Chen	ULINK Technology Inc.
Yoni Shternhell	Western Digital Technologies, Inc.

CONTENTS

List of Tables

1 Introduction

1.1 Document Purpose

This document provides examples of the communications between a host and a Storage Device implementing the TCG Storage Security Subsystem Class: Key Per I/O [3] and the TCG Storage Architecture Core Specification (Core Spec) [2] to perform the scenarios listed in section 2.

1.2 Scope and Intended Audience

The intended audience for this document is implementers of systems using TCG Key Per I/O SSC compliant Storage Devices.

1.3 Convention

1.3.1 Key Words

Key words are used to signify requirements.

The Key Words “SHALL”, “SHALL NOT”, “SHOULD,” and “MAY” are used in this document. These words are a subset of the RFC 2119 (see [1]) key words used by TCG. These key words are to be interpreted as described in [1].

In addition to the above key words, the following are also used in this document to describe the requirements of particular features, including tables, methods, and usages thereof:

- Mandatory (M): When a feature is Mandatory, the feature SHALL be implemented. A Compliance test SHALL validate that the feature is operational.
- Optional (O): When a feature is Optional, the feature MAY be implemented. If implemented, a Compliance test SHALL validate that the feature is operational.
- Excluded (X): When a feature is Excluded, the feature SHALL NOT be implemented. A Compliance test SHALL validate that the feature is not operational.
- Not Required (N): When a feature is Not Required, the feature MAY be implemented. No Compliance test is required.

1.3.2 Font Conventions

Names of methods and SP tables are in `Courier New font` (e.g., the `Set` method, the `Locking` table). This convention does not apply to method and table names appearing in headings or captions.

Hexadecimal numbers are in `Courier New font`.
All other text is in the Arial font.

1.3.3 Statement Types

All statements in this document are informative.

1.3.4 List Conventions

1.3.4.1 Lists Overview

Lists are associated with an introductory paragraph or phrase, and are numbered relative to that paragraph or phrase (i.e., all lists begin with an a) or 1) entry).

Each item in a list is preceded by an identification with the style of the identification being determined by whether the list is intended to be an ordered list or an unordered list.

If the item in a list is not a complete sentence, the first word in the item is not capitalized. If the item in a list is a complete sentence, the first word in the item is capitalized.

Each item in a list ends with a semicolon, except the last item, which ends in a period. The next to the last entry in the list ends with a semicolon followed by an “and” or an “or” (i.e., “…; and”, or “…; or”). The “and” is used if all the items in the list are required. The “or” is used if only one or more items in the list are required.

1.3.4.2 Unordered Lists

An unordered list is one in which the order of the listed items is unimportant (i.e., it does not matter where in the list an item occurs as all items have equal importance). Each list item shall start with a lowercase letter followed by a close parenthesis. If it is necessary to subdivide a list item further with an additional unordered list (i.e., have a nested unordered list), then the nested unordered list shall be indented and each item in the nested unordered list shall start with an uppercase letter followed by a close parenthesis.

The following is an example of an unordered list with a nested unordered list:

EXAMPLE - The following are the items for the assembly:
a) a box containing:
    A) a bolt;
    B) a nut; and
    C) a washer;
b) a screwdriver; and
c) a wrench.

1.3.4.3 Ordered Lists

An ordered list is one in which the order of the listed items is important (i.e., item n is required before item n+1). Each listed item starts with a Western-Arabic numeral followed by a close parenthesis. If it is necessary to subdivide a list item further with an additional unordered list (i.e., have a nested unordered list), then the nested unordered list shall be indented and each item in the nested unordered list shall start with an uppercase letter followed by a close parenthesis.

The following is an example of an ordered list with a nested unordered list:

EXAMPLE - The following are the instructions for the assembly:
1) remove the contents from the box;
2) assemble the item;
    A) use a screwdriver to tighten the screws; and
    B) use a wrench to tighten the bolts;
    and
3) take a break.

1.3.5 Numbering Conventions

A binary number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 and 1 immediately followed by a lowercase b (e.g., 0101b). Underscores or spaces may be included between characters in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b or 0_0101_1010b).

A hexadecimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 and/or the uppercase English letters A through F immediately preceded by “`0x`”. Underscores or spaces may be included between characters in hexadecimal number representations to increase readability or delineate field boundaries (e.g., `0xFD8C FA23` or `0x0B_FD8C_FA23`). Hexadecimal numbers are in `Courier New font`.

A decimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 not immediately followed by a lowercase b or lowercase h (e.g., 25). This standard uses the following conventions for representing decimal numbers:

a) the decimal separator (i.e., separating the integer and fractional portions of the number) is a period;
b) the thousands separator (i.e., separating groups of three digits in a portion of the number) is a space; and
c) the thousands separator is used in both the integer portion and the fraction portion of a number.

A decimal number represented in this standard with an overline over one or more digits following the decimal point is a number where the overlined digits are infinitely repeating (e.g., 666.$\overline{6}$ means 666.666 666… or 666 2/3, and 12.$\overline{142857}$ means 12.142 857 142 857… or 12 1/7).

1.3.6 Bit Conventions

Name (n:m), where n is greater than m, denotes a set of bits (e.g., Feature (7:0)).

1.3.7 Number Range Conventions

p..q, where p is less than q, represents a range of numbers (e.g., words 100..103 represents words 100, 101, 102, and 103).

1.3.8 Specify and Indicate Conventions

In a given message, command, or other information exchange between a requestor (e.g., a host) and a responder (e.g., an SD):

a) the requestor specifies information in the request; and
b) the responder indicates information in the response.

1.4 Document References

[1] IETF RFC 2119, 1997, “Key words for use in RFCs to Indicate Requirement Levels”. Available: https://datatracker.ietf.org/doc/html/rfc2119.

[2] Trusted Computing Group (TCG), “TCG Storage Architecture Core Specification”, Version 2.01. Available: https://trustedcomputinggroup.org.

[3] Trusted Computing Group (TCG), “TCG Storage Security Subsystem Class: Key Per I/O”, Version 1.00. Available: https://trustedcomputinggroup.org.

[4] OASIS, “OASIS Key Management Interoperability Protocol Specification”, Version 2.0. Available: https://www.oasis-open.org.

[5] OASIS, “OASIS Key Management Interoperability Protocol Usage Guide”, Version 2.0. Available: https://www.oasis-open.org.

[6] Trusted Computing Group (TCG), “Storage Interface Interactions Specification”, Version 1.11 Revision 1.18. Available: https://trustedcomputinggroup.org.

1.5 Definitions and Terms

Term	Definition
Eradicate	Irrevocably erase (e.g., cryptographically erase)
IF-RECV	An interface command used to retrieve security protocol data from the TPer (see [6])
IF-SEND	An interface command used to transmit security protocol data to the TPer (see [6])

Term	Definition
KEK	Key Encryption Key
Key Per I/O SP	A security provider described in the Key Per I/O SSC spec (see [3])
Key Per I/O SP is owned	A condition in which specific modifications of an SP have been made (see [6])
KMS	Key Management Server
KPIO	Key Per I/O
KTn	Key Tag number n
MEK	Media Encryption Key
NS	Namespace
NSn	Namespace number n
SD	Storage Device
SP	Security Provider
SSC	Security Subsystem Class. SSC specifications describe profiled sets of TCG functionality
TPer	The Trusted Peripheral is the subset of a SD for which TCG manages security

2 Disk Encryption using Key Per I/O SP

2.1 Overview

This document provides example communications with a device that complies with [3]. Examples are provided for the following scenarios:

- Discovering whether a Storage Device supports Key Per I/O SSC
- Taking ownership of the Storage Device
- Activating the Key Per I/O SP
- Changing the Admin1 PIN in the Key Per I/O SP
- Configuring Namespace to be Managed by Key Per I/O
- Unlocking Key Injection Interface
- Access Unlocking a KEK
- Associating KEK with Namespace
- Injecting plaintext KEK
- Replacing KEK with new wrapped KEK
- Injecting KEK wrapped with existing KEK
- Injecting KEK wrapped with Public Key
- Injecting MEK wrapped with KEK
- Clearing single MEK from TPer key cache
- Clearing all MEKs associated with Namespace from TPer key cache
- Reverting the Key Per I/O SP

3 Recommended Implementation

3.1 Overview

This section describes an example of the communications utilized in implementations of the use case scenarios, using commands described by the TCG Storage Architecture Core Specification [2] and the Key Per I/O SSC [3].

3.2 Brief Description of Discovery, Sessions, KMIP Messages, and Commands

3.2.1 Discovery

3.2.1.1 Discovering whether a Storage Device supports Key Per I/O SSC

This includes the sequence of operations that a host application should go through to determine whether a Storage Device supports the TCG Key Per I/O SSC specification [3].

3.2.1.1.1 Level 0 Discovery Request

IF_RECV with Protocol 01: Level 0 discovery (ComID `0x0001`)

A Key Per I/O SSC compliant Storage Device will return the following Level 0 Discovery response (additional descriptors may be present if the Storage Device supports multiple SSCs):

- Level 0 Discovery Header
- TPer Feature Descriptor
- Key Per I/O SSC Feature Descriptor

A Key Per I/O SSC compliant Storage Device may also return the following Level 0 Discovery response:

- Supported Data Removal Mechanism Feature Descriptor

3.2.1.2 Namespace Level0 Discovery

A Key Per I/O SSC compliant Storage Device will return the following Namespace Level 0 Discovery response:

- Namespace Level 0 Discovery Feature Descriptor

3.2.1.3 Exchange Communication Properties with a TPer

This section includes the sequence of operations that a host should follow to discover a Storage Device's communication properties and inform the Storage Device of the host's communication properties.

The `Properties` method may be used by a host to provide its communication properties to a TPer, and to retrieve the communication properties of the TPer. Configuring communication properties by the host through the `Properties` method is optional. Communication may occur using just the minimum communication capabilities which are configured by default.

3.2.2 Take Ownership of Storage Device

This section enumerates the steps a host follows to take ownership of a Storage Device (see section 3.3.4). The host:

1. Opens a session to the Admin SP as the Anybody authority

a. `StartSession`
b. `SyncSession`

2. Gets the MSID's PIN value from the `C_PIN` Table
   a. `Get` method
   b. `Get` Result
3. Closes the session
   a. End of Session
   b. End of Session Response
4. Opens a session to the Admin SP as the SID authority using the <MSID_password>
   a. `StartSession`
   b. `SyncSession`
5. Sets the <new_SID_password> value in the SID's `C_PIN` credential PIN column
   a. `Set` method
   b. `Set` Result
6. Closes the session
   a. End of Session
   b. End of Session Response

3.2.3 Activate Key Per I/O SP

This section enumerates the steps a host follows to `Activate` the Key Per I/O SP (see section 3.3.5). The host:

1. Opens a session to the Admin SP as the SID authority
   a. `StartSession`
   b. `SyncSession`
2. Determines the life cycle state of the Key Per I/O SP
   a. `Get` method
   b. `Get` Result
3. Activates the Key Per I/O SP by using the `Activate` method on the Key Per I/O SP object in the Admin SP
   a. `Activate` method
   b. `Activate` Result
4. Closes the session
   a. End of Session
   b. End of Session Response

3.2.4 Changing Admin1 PIN in Key Per I/O SP

This section enumerates the steps a host follows to change the Admin1 PIN in the Key Per I/O SP (see section 3.3.6). The host:

1. Opens a session to the Key Per I/O SP as Admin1

a. StartSession
b. SyncSession

2. Sets the <Admin1_password> value in Admin1's C_PIN credential PIN column
   a. Set method
   b. Set Result
3. Closes the session
   a. End of Session
   b. End of Session Response

3.2.5 Configure Namespace to be Managed by Key Per I/O

This section enumerates the steps a host follows to configure a Namespace to be managed by Key Per I/O (see section 3.3.7). This section is not applicable if Key Per I/O is configured to apply to all Namespaces in NVM subsystem (KPIOSC=1) [3]. A Namespace must be managed by Key Per I/O SP before MEKs or KEKs can be injected through the Key Injection Interface and associated with a Namespace. The host:

1. Opens a session to the Key Per I/O SP as Admin1
   a. StartSession
   b. SyncSession
2. Configures Namespace to be managed by Key Per I/O by setting the Managed column in the KeyTagAllocation Table to TRUE
   a. Set method
   b. Set response
3. Closes the session
   a. End of Session
   b. End of Session response

3.2.6 Enable Key Injection Interface

This section enumerates the steps a host follows to unlock the Key Injection Interface (see section 3.3.8). When the Key Injection Interface is in a locked state, MEKs and KEKs cannot be injected into the Storage Device through the Key Injection Interface.

After activating the Key Per I/O SP, the Key Injection Interface is in an unlocked state by default. The procedure that follows is necessary only when the KeyInjectionInterfaceLockEnabled and KeyInjectionInterfaceLocked columns in the KPIOPolicies Table are set to TRUE. These columns can be set to TRUE by either of the following actions:

a. configuring the KeyInjectionInterfaceLockEnabled and KeyInjectionInterfaceLocked columns in the KPIOPolicies Table to TRUE; or
b. configuring the KeyInjectionInterfaceLockEnabled column in the KPIOPolicies Table to TRUE and performing a TCG Power Cycle, which will transition the KeyInjectionInterfaceLocked column to TRUE.

The Key Injection Interface is set to the unlocked state when the host:

1. Opens a session to the Key Per I/O SP as Admin1
   a. StartSession

b. `SyncSession`

2. Unlocks the Key Injection Interface by setting the KeyInjectionInterfaceLocked column in the `KPIOPolicies` Table to FALSE
    a. `Set` method
    b. `Set` response
3. Close the session
    a. End of Session
    b. End of Session response

3.2.7 Unlock Access to KEK

This section enumerates the steps a host follows to unlock access to a KEK (see section 3.3.9). When a KEK is in an access locked state, a MEK or KEK that is wrapped with the access locked KEK cannot be injected into the Storage Device. The KEK must first be transitioned to an access unlocked state.

After activating the Key Per I/O SP, all KEKs are in an access unlocked state by default. The procedure that follows is necessary only when the AccessLockEnabled and AccessLocked column associated with a KEK are both set to TRUE. These columns can be set to TRUE by

a. configuring the AccessLockEnabled and AccessLocked columns to TRUE for the row in the KeyEncyrptionKey Table associated with the KEK; or
b. configuring the AccessLockEnabled column to TRUE and performing a TCG Power Cycle, which will transition the AccessLocked column to TRUE.

Note: After a MEK or KEK is injected into a TPer, the association between the wrapping KEK and the MEK/KEK is not tracked by the TPer. Therefore, if the wrapping KEK is transitioned to an access locked state after a MEK or KEK is injected into the TPer, this transition will not affect access to the MEK or KEK. For example, if MEK1 is wrapped with KEK1 and successfully injected into the TPer, MEK1 is unaffected by a subsequent transition of KEK1 into the access locked state.

The KEK is transitioned to the access unlocked state when the host:

1. Opens a session to the Key Per I/O SP as Admin1
    a. `StartSession`
    b. `SyncSession`
2. Unlocks access to a KEK by setting the AccessLocked column in the `KeyEncryptionKey` Table to FALSE
    a. `Set` method
    b. `Set` response
3. Close the session
    a. End of Session
    b. End of Session response

3.2.8 Associate KEK with Namespace

This section enumerates the steps a host follows to associate a KEK with a Namespace (see section 3.3.10). After activating the Key Per I/O SP, AllowedKeyEncryptionKeys column in the `KeyTagAllocation` Table is empty for

each Namespace. Before a MEK can be injected into a Storage Device for a particular Namespace, the KEK that wraps the MEK must be associated with the Namespace. The host:

1. Opens a session to the Key Per I/O SP as Admin1
    a. `StartSession`
    b. `SyncSession`
2. Associates the KEK with the Namespace by setting the AllowedKeyEncryptionKeys column in the `KeyTagAllocation` Table that is associated with the Namespace to include the KEK UID.
    a. `Set` method
    b. `Set` response
3. Closes the session
    a. End of Session
    b. End of Session response

3.2.9 Enable Plaintext KEK Provisioning for all KEKs

This section enumerates the steps a host follows to enable plaintext KEK provisioning for all KEKs in `KeyEncryptionKey` Table (see section 3.3.11). The host:

1. Opens a session to the Key Per I/O SP as Admin1
    a. `StartSession`
    b. `SyncSession`
2. Enable plaintext KEK provisioning for all KEKs in `KeyEncryptionKey` Table by setting the PlaintextKEKProgrammingEnabled column in the `KPIOPolicies` Table to TRUE.
    a. `Set` method
    b. `Set` response
3. Closes the session
    a. End of Session
    b. End of Session response

3.2.10 Enable Plaintext KEK Provisioning for Single KEK

This section enumerates the steps a host follows to enable plaintext KEK provisioning for a single KEK in `KeyEncryptionKey` Table (see section 3.3.12). The host:

1. Opens a session to the Key Per I/O SP as Admin1
    a. `StartSession`
    b. `SyncSession`
2. Enable plaintext KEK provisioning for KEK associated with KeyEncryptionKey1 by setting the AllowedKeyEncryptionKeys column in the `KeyEncryptionKey` Table associated with KeyEncryptionKey1 to include the NULLKeyEncryptionKey UID.

a. `Set` method
b. `Set` response

3. Closes the session
    a. End of Session
    b. End of Session response

3.2.11 Inject Plaintext KEK

This section enumerates the steps a host follows to provision a plaintext KEK into a TPer (see section 3.4.2). After activating Key Per I/O SP, the Key column in `KeyEncryptionKey` Table is empty for each KEK by default. If injecting a plaintext KEK is supported by the TPer, then the first KEK can be injected as plaintext. After a KEK is injected into the Storage Device, the KEK can be used to encrypt future KEKs (see section 3.2.12 and 3.2.13). A plaintext KEK can be injected into the Storage Device if any of the following are true:

a. plaintext provisioning is supported and the PlaintextKEKProgrammingEnabled column from the `KPIOPolicies` Table is configured to TRUE (see section 3.2.9). When the PlaintextKEKProgrammingEnabled column is set to TRUE, plaintext provisioning will be enabled for all rows in the `KeyEncryptionKey` Table;

b. plaintext provisioning is supported and the NULLKeyEncryptionKey UID is included in the AllowedKeyEncryptionKeys column of the `KeyEncryptionKey` Table associated with the KEK (see section 3.2.10). NULLKeyEncryptionKey UID has to be added to the AllowedKeyEncryptionKeys column for each row in the `KeyEncryptionKey` Table where plaintext provisioning is necessary. This gives the host finer grain control over which KEKs can be injected with plaintext provisioning; or

c. the Key column of the `KeyEncryptionKey` Table associated with the KEK is empty.

The host:

1. Unlocks the Key Injection Interface
    a. `Set` method
    b. `Set` response
2. Retrieves a plaintext KEK from the KMS
3. Injects the plaintext KEK into the Storage Device by performing a KMIP Import operation of Symmetric Key using Security Protocol `0x03`
    a. KMIP Import operation Request Message
    b. KMIP Import operation Response Message

3.2.12 Update KEK using Previously Provisioned KEK

This section enumerates the steps a host follows to overwrite an existing KEK in the `KeyEncryptionKey` Table with a new KEK. The new KEK is wrapped using a previously provisioned KEK associated with the same `KeyEncryptionKey` row and injected into the TPer (see section 3.4.3).

After activating the Key Per I/O SP, each AllowedKeyEncryptionKeys row in the `KeyEncyrptionKey` Table contains the KEK UID associated with the row. Therefore, a KEK can be used to update the `KeyEncryptionKey` Table row associated with the KEK by default.

To update a previously provisioned KEK, the host:

1. Unlocks the Key Injection Interface
    a. `Set` method
    b. `Set` response
2. Retrieves the plaintext KEK from the KMS
3. Injects the Plaintext KEK into the Storage Device by performing the KMIP Import operation of the Symmetric Key using Security Protocol `0x03`
    a. KMIP Import operation Request Message
    b. KMIP Import operation Response Message
4. Retrieves the new encrypted KEK and old KEK UID from KMS
5. Updates the previously provisioned KEK in the Storage Device by performing a KMIP Import operation of Symmetric Key using Security Protocol `0x03`
    a. KMIP Import operation Request Message
    b. KMIP Import operation Response Message

3.2.13 Inject KEK wrapped with Existing KEK

This section enumerates the steps a host follows to inject into a TPer a KEK that is wrapped with a previously provisioned KEK associated with a different `KeyEncryptionKey` row (see section 3.4.4). After activating Key Per I/O SP, a KEK cannot update a KEK associated with a different row in the `KeyEncryptionKey` Table by default. Before a KEK can update a KEK associated with a different row in the `KeyEncryptionKey` Table, the KEK UID must be added to the AllowedKeyEncryptionKeys column in the `KeyEncryptionKey` Table associated with the KEK that will be updated. The host:

1. Unlocks the Key Injection Interface
    a. `Set` method
    b. `Set` response
2. Retrieves the plaintext KEK from the KMS
3. Injects the Plaintext KEK into the Storage Device and associates the KEK with KeyEncryptionKey1 row in `KeyEncryptionKey` Table by performing a KMIP Import operation of Symmetric Key using Security Protocol `0x03`
4. Opens a session to the Key Per I/O SP as Admin1
    a. `StartSession`
    b. `SyncSession`
5. Configures KeyEncryptionKey2 in `KeyEncryptionKey` Table so the KEK injected into Storage Device using KeyEncryptionKey1 UID can be wrapped with the existing KEK associated with KeyEncryptionKey1. This is accomplished by adding the KeyEncryptionKey1 UID to the AllowedKeyEncryptionKeys column for KeyEncryptionKey2 row in the `KeyEncryptionKey` Table
    a. `Set` method
    b. `Set` response
6. Closes the session

a. End of Session
b. End of Session response

7. Retrieves the KEK wrapped with the KEK associated with KeyEncryptionKey1 and KEK UID that is associated with KeyEncryptionKey1 from the KMS
8. Injects the new wrapped KEK into the Storage Device and associates the KEK with KeyEncryptionKey2 by performing a KMIP Import operation of Symmetric Key using Security Protocol `0x03`
    a. KMIP Import operation Request Message
    b. KMIP Import operation Response Message

3.2.14 Enable KEK wrapping with PKI Public Key for all KEKs

This section enumerates the steps a host follows to enable PKI based KEK provisioning for all KEKs in `KeyEncryptionKey` Table (see section 3.3.13). In this scenario, the PKIProtectedKEKProgrammingEnabled column of the `KPIOPolicies` Table is set to TRUE to allow the host to inject any KEK protected with the Public Key. The host:

1. Opens a session to the Key Per I/O SP as Admin1
    a. `StartSession`
    b. `SyncSession`
2. Enables injection of any KEK wrapped with the Public Key by setting the PKIProtectedKEKProgrammingEnabled column in the `KPIOPolicies` Table to TRUE
    a. `Set` method
    b. `Set` response
3. Closes the session
    a. End of Session
    b. End of Session response

3.2.15 Enable KEK wrapping with PKI Public Key for Single KEK

This section enumerates the steps a host follows to enable PKI based KEK provisioning for a single KEK in `KeyEncryptionKey` Table (see section 3.3.14). The PKIPublicKeyEncryptionKey UID must be added to the AllowedKeyEncryptionKeys column in the `KeyEncryptionKey` Table associated with a KEK before the Public Key can protect the injected KEK. If the PKIProtectedKEKProgrammingEnabled column of the `KPIOPolicies` Table is set to TRUE (see section 3.2.14), then this procedure is not required since PKI based KEK provisioning is already enabled for all KEKs in the `KeyEncryptionKey` Table.

The host:

1. Opens a session to the Key Per I/O SP as Admin1
    a. `StartSession`
    b. `SyncSession`
2. Configures KeyEncryptionKey1 row so the KEK associated with the row can be wrapped with the PKI Public Key when the KEK is injected into the Storage Device. This is accomplished by adding the

PKIPublicKeyEncryptionKey UID to the AllowedKeyEncryptionKeys column associated with KeyEncryptionKey1

a. `Set` method

b. `Set` response

3. Closes the session

a. End of Session

b. End of Session response

3.2.16 Inject KEK wrapped with Public Key

This section enumerates the steps a host follows to inject a KEK into a Storage Device that is wrapped with a Public Key that is associated with the pre-provisioned Public Key Certificate (see section 3.4.5). If the PKIProtectedKEKProgrammingEnabled column of the `KPIOPolicies` Table is set to FALSE, then the PKIPublicKeyEncryptionKey UID must be added to the AllowedKeyEncryptionKeys column in the `KeyEncryptionKey` Table associated with a KEK before the Public Key can be used to protect the injected KEK. A PKI encrypted KEK can be injected into the Storage Device if:

a. PKI KEK provisioning is supported and the PKIProtectedKEKProgrammingEnabled column from the `KPIOPolicies` Table is configured to TRUE (see section 3.2.14); or

b. PKI KEK provisioning is supported and the PKIPublicKeyEncryptionKey UID is included in the AllowedKeyEncryptionKeys column of the `KeyEncryptionKey` Table associated with the KEK (see section 3.2.15).

After activating the Key Per I/O SP, the Key column in the `KeyEncryptionKey` Table is empty for each KEK by default. If injecting a KEK protected by Public Key encryption is supported by the TPer, then the first KEK can be injected wrapped with the Public Key. After a KEK is injected into the Storage Device, the KEK can be used to encrypted future KEKs (see section 3.2.12, 3.2.13, and 3.2.18).

The host:

1. Unlocks the Key Injection Interface

a. `Set` method

b. `Set` response

2. Opens a session to the Key Per I/O SP as Admin1

a. `StartSession`

b. `SyncSession`

3. Retrieves the UID of the `KeyPerIOPublicKeyCertificateData` byte table and the size of certificate data from `Certificates` Table

a. `Get` method

b. `Get` response

4. Retrieves the certificate data from the `KeyPerIOPublicKeyCertificateData` byte Table

a. `Get` method

b. `Get` response

5. Closes the session

a. End of Session
b. End of Session response

6. Imports the Certificate data into the KMS
7. Retrieves the new KEK wrapped with the Public Key associated with Certificate from KMS
8. Injects the new wrapped KEK into the Storage Device by performing a KMIP Import operation of Symmetric Key using Security Protocol `0x03`
    a. KMIP Import operation Request Message
    b. KMIP Import operation Response Message

3.2.17 Inject MEK wrapped with KEK

This section enumerates the steps a host follows to inject a MEK wrapped with a KEK into a Storage Device (see section 3.4.6). A MEK must be wrapped by a KEK when injecting the MEK into the Storage Device and only KEKs can be used to wrap MEKs. Before a wrapped MEK can be injected into the Storage Device, the KEK used to wrap the MEK must be added to the AllowedKeyEncryptionKeys column in the `KeyTagAllocation` Table for the Namespace the MEK will be associated.

The KEK used to wrap the MEK also must be in an access unlocked state (see section 3.2.7). The access locked state of the KEK that is used to wrap the MEK does not affect the MEK after the MEK is successfully injected into the Storage Device.

The host:

1. Unlocks the Key Injection Interface
    a. `Set` method
    b. `Set` response
2. Retrieves the KEK from the KMS
3. Injects the KEK into the Storage Device by performing a KMIP Import operation of Symmetric Key using Security Protocol `0x03`
4. Opens a session to the Key Per I/O SP as Admin1
    a. `StartSession`
    b. `SyncSession`
5. Configures the existing KEK so it can be used to unwrap the new wrapped MEK by setting the KEK UID associated with the KEK to AllowedKeyEncryptionKeys column in the `KeyTagAllocation` Table for the row associated with the Namespace where the MEK will be injected
    a. `Set` method
    b. `Set` response
6. Closes the session
    a. End of Session
    b. End of Session response
7. Retrieves the encrypted MEK from the KMS

8. Injects the wrapped MEK into the Storage Device by performing a KMIP Import operation of Symmetric Key using Security Protocol `0x03`
    a. KMIP Import operation Request Message
    b. KMIP Import operation Response Message

3.2.18 Inject KEK wrapped with KEK and Nonce

This section enumerates the steps a host follows to inject a KEK into the Storage Device, wrapped with a previously provisioned KEK and a Nonce retrieved from the TPer (see section 3.4.7). As described in section 3.2.13, the AllowedKeyEncryptionKeys column of the `KeyEncryptionKey` Table associated with the new KEK must include the wrapping KEK UID before injecting the new KEK. Also, support for replay protection is necessary to retrieve a Nonce from the TPer and inject a KEK wrapped with a Nonce (see section 3.3.2.1 for the process the host follows to discover replay protection support). If replay protection is supported by a TPer, then replay protection can be enabled by setting the ReplayProtectionEnabled column in the `KPIOPolicies` Table to TRUE.

Note: Configuring the ReplayProtectionEnabled column is not required each time a KEK is injected into the TPer since the ReplayProtectionEnabled column value is persistent across a TCG Power Cycle. The state of the replay protection feature can be obtained through in the Key Per I/O feature descriptor (see section 3.3.2.1).

The host:

1. Unlocks the Key Injection Interface
    a. `Set` method
    b. `Set` response
2. Opens a session to the Key Per I/O SP as Admin1
    a. `StartSession`
    b. `SyncSession`
3. Enables replay protection when injecting MEKs and KEKs by setting the ReplayProtectionEnabled column in the `KPIOPolicies` Table to TRUE.
    a. `Set` method
    b. `Set` response
4. Configures KeyEncryptionKey2 in `KeyEncryptionKey` Table so the KEK injected into Storage Device using KeyEncryptionKey1 UID can be wrapped with an existing KEK associated with KeyEncryptionKey1. This is accomplished by adding the KeyEncryptionKey1 UID to the AllowedKeyEncryptionKeys column for KeyEncryptionKey2 row in the `KeyEncryptionKey` Table
    a. `Set` method
    b. `Set` response
5. Closes the session
    a. End of Session
    b. End of Session response
6. Retrieves the Nonce from the TPer
    a. Get Nonce Command
7. Requests the KMS to generate a new KEK and wraps the KEK using the Nonce retrieved from the TPer

8. Retrieves the encrypted KEK from the KMS
9. Injects the wrapped KEK with Nonce by performing a KMIP Import operation of Symmetric Key using Security Protocol `0x03`
    a. KMIP Import operation Request Message
    b. KMIP Import operation Response Message

3.2.19 Clear Single MEK from Single Key Tag Slot

This section enumerates the steps a host follows to clear a single MEK from a TPer's key cache (see section 3.3.15). The host:

1. Opens a session to the Key Per I/O SP as Admin1
    a. `StartSession`
    b. `SyncSession`
2. Enables a Clear Single MEK Command by setting the ClearSingleMEKAllowed column in the `KPIOPolicies` Table to FALSE.
    a. `Set` method
    b. `Set` response
3. Closes the session
    a. End of Session
    b. End of Session response
4. Clears one MEK from the TPer's key cache by sending the Clear Single MEK command with the NSID and Key Tag associated with the MEK
    a. Clear Single MEK Command request
    b. Clear Single MEK Command response

3.2.20 Clear All MEKs in Namespace

This section enumerates the steps a host follows to clear all MEKs from a TPer's key cache that is associated with a Namespace (see section 3.3.16). The host:

1. Opens a session to the Key Per I/O SP as Admin1
    a. `StartSession`
    b. `SyncSession`
2. Enable Clear All MEKs Command by setting the ClearAllMEKsAllowed column in the `KPIOPolicies` Table to FALSE.
    a. `Set` method
    b. `Set` response
3. Closes the session
    a. End of Session
    b. End of Session response

4. Clears all MEKs from the TPer's key cache associated with a Namespace by sending the Clear All MEK command with the associated NSID
    a. Clear All MEKs Command request
    b. Clear All MEKs Command response

3.2.21 Revert Key Per I/O SP

This section enumerates the steps a host follows to revert a TPer to its Original Factory State (see section 3.3.17). After the TPer is successfully reverted, all KEKs and MEKs injected into the Storage Device through the Key Injection Interface will no longer be accessible. If the Key Per I/O SP is subsequently activated, previously injected KEKs and MEKs will not be accessible. The host:

1. Opens a session to the Admin SP as SID
    a. `StartSession`
    b. `SyncSession`
2. Reverts the Admin SP
    a. `Revert`
    b. `Revert` Result

3.2.22 Revert TPer

This section enumerates the steps a host follows to revert a Key Per I/O SP to its Original Factory State (see section 3.3.18). After the Key Per I/O SP is successfully reverted, all KEKs and MEKs injected into the Storage Device through the Key Injection Interface will no longer be accessible. If the Key Per I/O SP is subsequently activated, previously injected KEKs and MEKs will not be accessible. The host:

1. Opens a session to the Admin SP as SID
    a. `StartSession`
    b. `SyncSession`
2. Reverts the Key Per I/O SP
    a. `Revert`
    b. `Revert` Result

3.3 TCG Command Tokenization

3.3.1 Overview

This section provides additional details about the commands described in section 3.2, such as the tokenization of each command and the packaging of those commands in Subpackets, Packets, and ComPackets.

The following is a list of command parameters that vary in actual implementations. In this section, the values chosen for these parameters are as follows:

1. All commands use a statically allocated Vendor Unique ComID value of `0x08000000`.
2. The host always uses the Host Session Number (HSN) `0x00000001`.

3. The TPer always uses the TPer Session Number (TSN) `0x00001001`.
4. Communications sent from the host to the TPer have a Packet.SeqNumber of 0.
5. Communications sent from the TPer to the Host have a Packet.SeqNumber of 0.

All transfers between the host and storage device are in 512-byte blocks. If the ComPacket does not end at a 512-byte boundary, bytes of `0x00` are appended after the ComPacket to pad the buffer to a 512-byte alignment.

3.3.1.1 TCG Method Signature

The TCG method signature pseudo-code used in this document is a customized version of the TCG method signature pseudo-code used in the TCG Core Specification [2]. The method signature used in the TCG Core Specification combines the method invocation and result in one signature. In this document, the formatting of the method signatures is the same except the method invocation and the result are split into two method signatures.

The method signature representing the method invocation follows the format:

```
session[TSN:HSN] -> InvokingID.MethodName[Parameters]
```

The string "->" in the method signature is used to indicate that the pseudo-code represents a host sending a method invocation to a TPer. The TSN is the TPer session number and the HSN is the host session number, which are extracted from SyncSession (see section 3.3.3.2).

The method signature representing the result of the method invocation has two formats. If the response is a TCG method, then the signature is:

```
session[TSN:HSN] <- InvokingID.MethodName[Result]
```

Otherwise, the method signature follows:

```
session[TSN:HSN] <- [Result]
```

The string "<-" in the method signature is used to indicate that the pseudo-code represents the TPer response to the method invocation. Result may be empty.

3.3.2 Discovery

3.3.2.1 Level 0 Discovery

The values in the Level 0 Discovery Response reported in this section are examples and may vary between implementations.

3.3.2.1.1 Response

The following example is the response payload for a Level 0 Discovery request from a host. The Level 0 Discovery response contains the following feature descriptors: Level 0 Discovery Header, TPer Feature Descriptor, and Key Per I/O SSC Feature Descriptor. These feature descriptors will be present in the response payload from a Storage Device that supports Key Per I/O SP. The response may also include the optional Supported Data Removal Mechanism Feature Descriptor.

Security Protocol `0x01` ComPacket Payload – TCG byte encoding of Level 0 Discovery response (see Table 1 for a description of the following byte encoding):

```
0000    00 00 00 90 00 00 00 01  00 00 00 00 00 00 00 00
0010    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0030    00 01 10 0c 11 00 00 00  00 00 00 00 00 00 00 00
0040    03 05 10 2c 08 00 00 01  08 01 00 01 00 00 00 01
0050    12 00 00 01 00 07 00 02  00 01 00 03 00 00 00 00
0060    00 00 00 02 00 00 00 02  00 01 10 00 00 00 00 00
0070    04 04 10 20 00 00 01 00  00 00 00 00 00 00 00 00
0080    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 1: Level 0 Discovery Response

Bytes (Hex)	Purpose	Value	Notes
Level 0 Discovery Response			
Level 0 Discovery Header			
00 00 00 90	Length Of Parameter Data	144	This field is vendor unique
00 00 00 01	Data Structure Revision	0x00000001	0x00000001 or any version that supports the defined features in this SSC
00 00 00 00 00 00 00 00	Reserved	0	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	Vendor Specific	0	This field is vendor unique
TPer Feature Descriptor			
00 01	Feature Code	0x0001	
10	Version	0x10	(7:4) Version = 0x1 (Note: other values may be present) (3:0) Reserved
0C	Length	0x0C	12 bytes
11	TPer Feature Supported	0x11	Sync Supported and Streaming Supported Note: This field is vendor unique - other values may be present
00 00 00 00 00 00 00 00 00 00 00 00	Reserved	0	
Key Per I/O SSC Feature Descriptor			
03 05	Feature Code	0x0305	

Bytes (Hex)	Purpose	Value	Notes
10	Feature Descriptor Version Number	0x10	(7:4) Feature Descriptor Version Number = 0x1 (4:0) SSC Minor Version Number = TCG Key Per I/O SSC Specification v1.00
2C	Length	0x2C	
08 00	Protocol 0x01 Base ComID	0x0800	This field is vendor unique - other values may be present
00 01	Protocol 0x01 Number ComID Supported	1	
08 01	Protocol 0x03 Base ComID	0x0801	This field is vendor unique - other values may be present
00 01	Protocol 0x03 Number ComID Supported	1	
00	Initial C_PIN_SID PIN Indicator	0	This field is vendor unique - other values may be present
00	Behavior of C_PIN_SID PIN upon TPer Revert	0	This field is vendor unique - other values may be present
00 01	Number of Key Per IO SP Admin Authorities Supported	1	
13	Key Per IO Features	0x12	Key Per I/O Disabled – other values may be present Key Per I/O Scope applies to all Namespaces Replay Protection Supported
00 00	Maximum Supported Key Unique Identifier Length	0	This field is vendor unique - other values may be present
01 00	Key Injection Protocol Formats	0x0100	KMIP Formatted Key Injection Supported
07 00	Wrapping Algorithms	0x0700	NIST AES-KW Supported NIST AES-GCM Supported NIST RSA-OAEP Supported
02 00	AES Key Wrapping Key Sizes	0x0200	AES-256 Wrapping Key Supported
01 00	RSA OAEP Key Wrapping Key Sizes	0x0100	RSA2K Wrapping Key Supported

Bytes (Hex)	Purpose	Value	Notes
03 00	KEK Provisioning Models	0x0300	Plaintext KEK Provisioning Supported PKI-based KEK Transport Supported
00 00 00	Reserved	0	
00 00 00 02	Number of Key Encryption Keys Supported	2	Two KEK Supported
00 00 00 02	Total Number of Key Tags Supported	2	Two Key Tag Supported
00 01	Maximum Number of Key Tags Supported Per Namespace	1	One Key Tag per Namespace Supported
10	Get Nonce Command Nonce Length	16	16 Byte Nonce Length
00 00 00 00 00	Reserved	0	
Supported Data Removal Mechanism Feature Descriptor			
04 04	Feature Code	0x0404	
10	Version	0x10	(7:4) Version = 0x1 (Note: other values may be present) (3:0) Reserved
20	Length	0x20	
00	Reserved	0	
00	Data Removal Operation Flag	0	
01	Supported Data Removal Mechanism	0x01	Overwrite Data Erase
00	Data Removal Time Format Flag	0	
00 00	Data Removal Time for Bit 0	0	
00 00	Data Removal Time for Bit 1	0	
00 00	Data Removal Time for Bit 2	0	
00 00	Data Removal Time for Bit 3	0	

Bytes (Hex)	Purpose	Value	Notes
00 00	Data Removal Time for Bit 4	0	
00 00	Data Removal Time for Bit 5	0	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	Reserved	0	

3.3.2.2 Namespace Level 0 Discovery

The values in the Namespace Level 0 Discovery Response reported in this section are examples and may vary between implementations. The Namespace Level 0 Discovery Response is retrieved for a particular Namespace by selecting the NamespaceID in the IF_RECV command.

3.3.2.2.1 Response

Security Protocol 0x01 ComPacket Payload – TCG byte encoding of Namespace Level 0 Discovery response (see Table 2 for a description of the following byte encoding):

```
0000    00 00 00 4c 00 00 00 01  00 00 00 00 00 00 00 00
0010    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0030    04 0a 10 1c 01 00 00 00  00 00 00 00 00 00 00 00
0040    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 2: Namespace Level 0 Discovery Response

Bytes (Hex)	Purpose	Value	Notes
Level 0 Discovery Response			
Namespace Level 0 Discovery Header			
00 00 00 4C	Length of Parameter Data	76	
00 00 00 01	Data Structure Revision	1	Revision = 0x00000001
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	Reserved	0	
Namespace Key Per I/O Capabilities Feature Descriptor			
04 0A	Feature Code	0x040A	

Bytes (Hex)	Purpose	Value	Notes
10	Version	0x10	(7:4) Version = 0x1 (Note: other values may be present) (3:0) Reserved
1C	Length	0x1C	28 bytes
01	Feature flags	0x01	Managed By Key Per I/O
00 00	Number of Allocated Key Tags	0	No Key Tags allocated for Namespace
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	Reserved	0	

3.3.2.3 Exchange Communication Properties

In the following example, a host informs a TPer of the host's communication properties through the Properties method:

- Its Security Protocol 0x01 and 0x02 receive buffer is 4KB in size (MaxComPacketSize)
- Its Security Protocol 0x01 and 0x02 transmit buffer is 4KB in size (MaxResponseComPacketSize)
- It supports receiving Packets up to 4076 bytes in size (MaxPacketSize)
- It supports individual tokens up to 4040 bytes in size (MaxIndTokenSize)
- It supports a maximum of one Packet per ComPacket (MaxPackets)
- It supports a maximum of one Subpacket per Packet (MaxSubpackets)
- It supports a maximum of one method invocation per Subpacket (MaxMethods)
- Its Security Protocol 0x03 receive buffer is 4KB in size (Protocol3MaxPayloadSize)
- It supports a maximum of two KMIP Batch Items per ComPacket (Protocol3MaxKmipBatchItems)

In the Properties response, a TPer informs a host of the TPer's communication properties:

- Its Security Protocol 0x01 and 0x02 receive buffer is 8KB in size (MaxComPacketSize)
- Its Security Protocol 0x01 and 0x02 transmit buffer is 8KB in size (MaxResponseComPacketSize)
- It supports receiving Packets up to 8172 bytes in size (MaxPacketSize)
- It supports individual tokens up to 8136 bytes in size (MaxIndTokenSize)
- It supports a maximum of one Packet per ComPacket (MaxPackets)
- It supports a maximum of one Subpacket per Packet (MaxSubpackets)
- It supports a maximum of one method invocation per Subpacket (MaxMethods)
- It does not accept continued tokens (ContinuedTokens)
- It does not support Packet sequence numbers (SequenceNumbers)
- It does not support the Packet ACK/NAK protocol (AckNak)

- It does not support the Asynchronous Communication Protocol (Asynchronous)
- It supports a maximum of one simultaneous session (MaxSessions)
- It supports a maximum of two authentications within a session (MaxAuthentications)
- It supports a maximum of one level of transactions within a session (MaxTransactionLimit)
- It supports a default session timeout of two minutes
- It will send ComPackets no larger than 4KB (HostProperties MaxComPacketSize)
- It will send Packets no larger than 4076 bytes (HostProperties MaxPacketSize)
- It will send individual tokens no larger than 4040 bytes (HostProperties MaxIndTokenSize)
- It will send no more than one Packet per ComPacket (HostProperties MaxPackets)
- It will send no more than one Subpacket per Packet (HostProperties MaxSubpackets)
- It will send no more than one method invocation per Subpacket (HostProperties MaxMethods)
- Its Security Protocol `0x03` receive buffer is 4KB in size (Protocol3MaxPayloadSize)
- It supports a maximum of two KMIP Batch Items per ComPacket (Protocol3MaxKmipBatchItems)

Note that in this example, the TPer does not make use of the MaxResponseComPacketSize property for HostProperties, and therefore that property is not echoed back in the HostProperties parameter of the Properties response.

3.3.2.3.1 Host to TPer Properties invocation

In this example, the Protocol3MaxKmipBatchItems Host Property is set to two to support the MEK injection section 3.4.5.7.1 since two KMIP Batch Items are necessary inject an XTS-AES-256 MEK.
The following pseudo-code is the signature of the `Properties` method used to set the host properties listed in section 3.3.2.3:

```
session[0:0] -> SMUID.Properties[ HostProperties = [“MaxComPacketSize” = 4096,
“MaxResponseComPacketSize” = 4096, “MaxPacketSize” = 4076, “MaxIndTokenSize” = 4040,
“MaxPackets” = 1, “MaxSubpackets” = 1, “MaxMethods” = 1, “Protocol3MaxPayloadSize” =
4096, “Protocol3MaxKmipBatchItems” = 2] ]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding of `Properties` method (see Table 3 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 01 0c 00 00 00 00  00 00 00 00 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 f4 00 00 00 00
0030    00 00 00 00 00 00 00 e8  f8 a8 00 00 00 00 00 00
0040    00 ff a8 00 00 00 00 00  00 ff 01 f0 f2 00 f0 f2
0050    d0 10 4d 61 78 43 6f 6d  50 61 63 6b 65 74 53 69
0060    7a 65 82 10 00 f3 f2 d0  18 4d 61 78 52 65 73 70
0070    6f 6e 73 65 43 6f 6d 50  61 63 6b 65 74 53 69 7a
0080    65 82 10 00 f3 f2 ad 4d  61 78 50 61 63 6b 65 74
0090    53 69 7a 65 82 0f ec f3  f2 af 4d 61 78 49 6e 64
00a0    54 6f 6b 65 6e 53 69 7a  65 82 0f c8 f3 f2 aa 4d
00b0    61 78 50 61 63 6b 65 74  73 01 f3 f2 ad 4d 61 78
00c0    53 75 62 50 61 63 6b 65  74 73 01 f3 f2 aa 4d 61
00d0    78 4d 65 74 68 6f 64 73  01 f3 f2 d0 17 50 72 6f
00e0    74 6f 63 6f 6c 33 4d 61  78 50 61 79 6c 6f 61 64
00f0    53 69 7a 65 82 10 00 f3  f2 d0 1a 50 72 6f 74 6f
```

```
0100    63 6f 6c 33 4d 61 78 4b  6d 69 70 42 61 74 63 68
0110    49 74 65 6d 73 02 f3 f1  f3 f1 f9 f0 00 00 00 f1
0120    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 3: Properties Method

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 01 0C	Length	268	Number of bytes in Packet
Packet			
00 00 00 00	TPerSN	0	TPer Session Number
00 00 00 00	HostSN	0	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 F4	Length	244	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 E8	Length	232	Number of bytes in Payload
Payload			
F8	Call Token		Begin Properties Method
A8 00 00 00 00 00 00 00 FF	Short Atom Token	SessionManager_UID	Session Manager UID

Bytes (Hex)	Purpose	Value	Notes
A8 00 00 00 00 00 00 FF 01	Short Atom Token	PropertiesMethod_UID	Properties Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin HostProperties Name-Value pair
00	Tiny Atom Token	0	HostProperties name
F0	Start List Token		Begin HostProperties Name-Value list
F2	Start Name Token		Begin Name-Value pair
D0 10 4D 61 78 43 6F 6D 50 61 63 6B 65 74 53 69 7A 65	Medium Atom Token	MaxComPacketSize	MaxComPacketSize name
82 10 00	Short Atom Token	4096	MaxComPacketSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
D0 18 4D 61 78 52 65 73 70 6F 6E 73 65 43 6F 6D 50 61 63 6B 65 74 53 69 7A 65	Medium Atom Token	MaxResponseComPacketSize	MaxResponseComPacketSize name
82 10 00	Short Atom Token	4096	MaxResponseComPacketSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AD 4D 61 78 50 61 63 6B 65 74 53 69 7A 65	Short Atom Token	MaxPacketSize	MaxPacketSize name
82 0F EC	Short Atom Token	4076	MaxPacketSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AF 4D 61 78 49 6E 64 54 6F 6B 65 6E 53 69 7A 65	Short Atom Token	MaxIndTokenSize	MaxIndTokenSize name
82 0F C8	Short Atom Token	4040	MaxIndTokenSize value

Bytes (Hex)	Purpose	Value	Notes
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AA 4D 61 78 50 61 63 6B 65 74 73	Short Atom Token	MaxPackets	MaxPackets name
01	Tiny Atom Token	1	MaxPackets value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AD 4D 61 78 53 75 62 50 61 63 6B 65 74 73	Short Atom Token	MaxSubPackets	MaxSubPackets name
01	Tiny Atom Token	1	MaxSubPackets value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AA 4D 61 78 4D 65 74 68 6F 64 73	Short Atom Token	MaxMethods	MaxMethods name
01	Tiny Atom Token	1	MaxMethods value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
D0 17 50 72 6F 74 6F 63 6F 6C 33 4D 61 78 50 61 79 6C 6F 61 64 53 69 7A 65	Medium Atom Token	Protocol3MaxPayloadSize	Protocol3MaxPayloadSize name
82 10 00	Short Atom Token	4096	Protocol3MaxPayloadSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
D0 1A 50 72 6F 74 6F 63 6F 6C 33 4D 61 78 4B 6D 69 70 42 61 74 63 68 49 74 65 6D 73	Medium Atom Token	Protocol3MaxKmipBatchItems	Protocol3MaxKmipBatchItems name
02	Tiny Atom Token	2	Protocol3MaxKmipBatchItems value

Bytes (Hex)	Purpose	Value	Notes
F3	End Name Token		End Name-Value pair
F1	End List Token		End HostProperties Name-Value list
F3	End Name Token		End HostProperties Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

3.3.2.3.2 TPer to Host Properties response

The following pseudo-code is the signature of the `Properties` method response that contains the host properties and TPer properties listed in section 3.3.2.3:

```
session[0:0] <- SMUID.Properties[ Properties : [“MaxComPacketSize” = 8192,
“MaxResponseComPacketSize” = 8192, “MaxPacketSize” = 8172, “MaxIndTokenSize” = 8136,
“MaxPackets” = 1, “MaxSubpackets” = 1, “MaxMethods” = 1, “Protocol3MaxPayloadSize” =
4096, “Protocol3MaxKmipBatchItems” = 2, “ContinuedTokens” = FALSE, “SequenceNumbers”
= FALSE, “AckNak” = FALSE, “Asynchronous” = FALSE, “MaxSessions” = 1,
“MaxAuthentications” = 2, “MaxTransactionLimit” = 1, “DefSessionTimeout” = 120000],
HostProperties = [“MaxComPacketSize” = 4096, “MaxPacketSize” = 4076,
“MaxIndTokenSize” = 4040, “MaxPackets” = 1, “MaxSubpackets” = 1, “MaxMethods” = 1,
“Protocol3MaxPayloadSize” = 4096, “Protocol3MaxKmipBatchItems” = 2] ]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding of `Properties` method response (see Table 4 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 02 50 00 00 00 00  00 00 00 00 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 02 38 00 00 00 00
0030    00 00 00 00 00 00 02 2a  f8 a8 00 00 00 00 00 00
0040    00 ff a8 00 00 00 00 00  00 ff 01 f0 f0 f2 d0 10
0050    4d 61 78 43 6f 6d 50 61  63 6b 65 74 53 69 7a 65
0060    82 10 00 f3 f2 d0 18 4d  61 78 52 65 73 70 6f 6e
0070    73 65 43 6f 6d 50 61 63  6b 65 74 53 69 7a 65 82
0080    20 00 f3 f2 ad 4d 61 78  50 61 63 6b 65 74 53 69
0090    7a 65 82 1f ec f3 f2 af  4d 61 78 49 6e 64 54 6f
00a0    6b 65 6e 53 69 7a 65 82  1f c8 f3 f2 aa 4d 61 78
00b0    50 61 63 6b 65 74 73 01  f3 f2 ad 4d 61 78 53 75
00c0    62 50 61 63 6b 65 74 73  01 f3 f2 aa 4d 61 78 4d
00d0    65 74 68 6f 64 73 01 f3  f2 d0 17 50 72 6f 74 6f
```

```
00e0    63 6f 6c 33 4d 61 78 50  61 79 6c 6f 61 64 53 69
00f0    7a 65 82 10 00 f3 f2 d0  1a 50 72 6f 74 6f 63 6f
0100    6c 33 4d 61 78 4b 6d 69  70 42 61 74 63 68 49 74
0110    65 6d 73 02 f3 f2 af 43  6f 6e 74 69 6e 75 65 64
0120    54 6f 6b 65 6e 73 00 f3  f2 af 53 65 71 75 65 6e
0130    63 65 4e 75 6d 62 65 72  73 00 f3 f2 a6 41 63 6b
0140    4e 61 6b 00 f3 f2 ac 41  73 79 6e 63 68 72 6f 6e
0150    6f 75 73 00 f3 f2 ab 4d  61 78 53 65 73 73 69 6f
0160    6e 73 01 f3 f2 d0 12 4d  61 78 41 75 74 68 65 6e
0170    74 69 63 61 74 69 6f 6e  73 02 f3 f2 d0 13 4d 61
0180    78 54 72 61 6e 73 61 63  74 69 6f 6e 4c 69 6d 69
0190    74 01 f3 f2 d0 11 44 65  66 53 65 73 73 69 6f 6e
01a0    54 69 6d 65 6f 75 74 83  01 d4 c0 f3 f1 f2 00 f0
01b0    f2 d0 10 4d 61 78 43 6f  6d 50 61 63 6b 65 74 53
01c0    69 7a 65 82 10 00 f3 f2  ad 4d 61 78 50 61 63 6b
01d0    65 74 53 69 7a 65 82 0f  ec f3 f2 af 4d 61 78 49
01e0    6e 64 54 6f 6b 65 6e 53  69 7a 65 82 0f c8 f3 f2
01f0    aa 4d 61 78 50 61 63 6b  65 74 73 01 f3 f2 ad 4d
0200    61 78 53 75 62 50 61 63  6b 65 74 73 01 f3 f2 aa
0210    4d 61 78 4d 65 74 68 6f  64 73 01 f3 f2 d0 17 50
0220    72 6f 74 6f 63 6f 6c 33  4d 61 78 50 61 79 6c 6f
0230    61 64 53 69 7a 65 82 10  00 f3 f2 d0 1a 50 72 6f
0240    74 6f 63 6f 6c 33 4d 61  78 4b 6d 69 70 42 61 74
0250    63 68 49 74 65 6d 73 02  f3 f1 f3 f1 f9 f0 00 00
0260    00 f1 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0270    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
03f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 4: Properties Method Response

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 02 50	Length	592	Number of bytes in Packet
Packet			
00 00 00 00	TPerSN	0	TPer Session Number
00 00 00 00	HostSN	0	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00	Acknowledgement	0	
00 00 02 38	Length	568	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 02 2A	Length	554	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 00 00 00 00 FF	Short Atom Token	SessionManager_UID	Session Manager UID
A8 00 00 00 00 00 00 FF 01	Short Atom Token	PropertiesMethod_UID	Properties Method UID
F0	Start List Token		Begin method parameter list
F0	Start List Token		Begin TPer Properties list
F2	Start Name Token		Begin Name-Value pair
D0 10 4D 61 78 43 6F 6D 50 61 63 6B 65 74 53 69 7A 65	Medium Atom Token	MaxComPacketSize	MaxComPacketSize name
82 10 00	Short Atom Token	4096	MaxComPacketSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
D0 18 4D 61 78 52 65 73 70 6F 6E 73 65 43 6F 6D 50 61 63 6B 65 74 53 69 7A 65	Medium Atom Token	MaxResponseComPacketSize	MaxResponseComPacketSize name
82 20 00	Short Atom Token	8192	MaxResponseComPacketSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair

Bytes (Hex)	Purpose	Value	Notes
AD 4D 61 78 50 61 63 6B 65 74 53 69 7A 65	Short Atom Token	MaxPacketSize	MaxPacketSize name
82 1F EC	Short Atom Token	8172	MaxPacketSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AF 4D 61 78 49 6E 64 54 6F 6B 65 6E 53 69 7A 65	Short Atom Token	MaxIndTokenSize	MaxIndTokenSize name
82 1F C8	Short Atom Token	8136	MaxIndTokenSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AA 4D 61 78 50 61 63 6B 65 74 73	Short Atom Token	MaxPackets	MaxPackets name
01	Tiny Atom Token	1	MaxPackets value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AD 4D 61 78 53 75 62 50 61 63 6B 65 74 73	Short Atom Token	MaxSubPackets	MaxSubPackets name
01	Tiny Atom Token	1	MaxSubPackets value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AA 4D 61 78 4D 65 74 68 6F 64 73	Short Atom Token	MaxMethods	MaxMethods name
01	Tiny Atom Token	1	MaxMethods value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
D0 17 50 72 6F 74 6F 63 6F 6C 33 4D 61 78 50 61 79 6C 6F 61 64 53 69 7A 65	Medium Atom Token	Protocol3MaxPayloadSize	Protocol3MaxPayloadSize name

Bytes (Hex)	Purpose	Value	Notes
82 10 00	Short Atom Token	4096	Protocol3MaxPayloadSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
D0 1A 50 72 6F 74 6F 63 6F 6C 33 4D 61 78 4B 6D 69 70 42 61 74 63 68 49 74 65 6D 73	Medium Atom Token	Protocol3MaxKmipBatchItems	Protocol3MaxKmipBatchItems name
02	Tiny Atom Token	2	Protocol3MaxKmipBatchItems value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AF 43 6F 6E 74 69 6E 75 65 64 54 6F 6B 65 6E 73	Short Atom Token	ContinuedTokens	ContinuedTokens name
00	Tiny Atom Token	0	ContinuedTokens value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AF 53 65 71 75 65 6E 63 65 4E 75 6D 62 65 72 73	Short Atom Token	SequenceNumbers	SequenceNumbers name
00	Tiny Atom Token	0	SequenceNumbers value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
A6 41 63 6B 4E 61 6B	Short Atom Token	AckNak	AckNak name
00	Tiny Atom Token	0	AckNak value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AC 41 73 79 6E 63 68 72 6F 6E 6F 75 73	Short Atom Token	Asynchronous	Asynchronous name
00	Tiny Atom Token	0	Asynchronous value

Bytes (Hex)	Purpose	Value	Notes
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AB 4D 61 78 53 65 73 73 69 6F 6E 73	Short Atom Token	MaxSessions	MaxSessions name
01	Tiny Atom Token	1	MaxSessions value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
D0 12 4D 61 78 41 75 74 68 65 6E 74 69 63 61 74 69 6F 6E 73	Medium Atom Token	MaxAuthentications	MaxAuthentications name
02	Tiny Atom Token	2	MaxAuthentications value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
D0 13 4D 61 78 54 72 61 6E 73 61 63 74 69 6F 6E 4C 69 6D 69 74	Medium Atom Token	MaxTransactionLimit	MaxTransactionLimit name
01	Tiny Atom Token	1	MaxTransactionLimit value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
D0 11 44 65 66 53 65 73 73 69 6F 6E 54 69 6D 65 6F 75 74	Medium Atom Token	DefSessionTimeout	DefSessionTimeout name
83 01 D4 C0	Short Atom Token	120000	DefSessionTimeout value
F3	End Name Token		End Name-Value pair
F1	End List Token		End TPer Properties list
F2	Start Name Token		Begin HostProperties Name-Value pair
00	Tiny Atom Token	0	HostProperties name
F0	Start List Token		Begin HostProperties list
F2	Start Name Token		Begin Name-Value pair

Bytes (Hex)	Purpose	Value	Notes
D0 10 4D 61 78 43 6F 6D 50 61 63 6B 65 74 53 69 7A 65	Medium Atom Token	MaxComPacketSize	MaxComPacketSize name
82 10 00	Short Atom Token	4096	MaxComPacketSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AD 4D 61 78 50 61 63 6B 65 74 53 69 7A 65	Short Atom Token	MaxPacketSize	MaxPacketSize name
82 0F EC	Short Atom Token	4076	MaxPacketSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AF 4D 61 78 49 6E 64 54 6F 6B 65 6E 53 69 7A 65	Short Atom Token	MaxIndTokenSize	MaxIndTokenSize name
82 0F C8	Short Atom Token	4040	MaxIndTokenSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AA 4D 61 78 50 61 63 6B 65 74 73	Short Atom Token	MaxPackets	MaxPackets name
01	Tiny Atom Token	1	MaxPackets value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AD 4D 61 78 53 75 62 50 61 63 6B 65 74 73	Short Atom Token	MaxSubPackets	MaxSubPackets name
01	Tiny Atom Token	1	MaxSubPackets value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
AA 4D 61 78 4D 65 74 68 6F 64 73	Short Atom Token	MaxMethods	MaxMethods name
01	Tiny Atom Token	1	MaxMethods value

Bytes (Hex)	Purpose	Value	Notes
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
D0 17 50 72 6F 74 6F 63 6F 6C 33 4D 61 78 50 61 79 6C 6F 61 64 53 69 7A 65	Medium Atom Token	Protocol3MaxPayloadSize	Protocol3MaxPayloadSize name
82 10 00	Short Atom Token	4096	Protocol3MaxPayloadSize value
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
D0 1A 50 72 6F 74 6F 63 6F 6C 33 4D 61 78 4B 6D 69 70 42 61 74 63 68 49 74 65 6D 73	Medium Atom Token	Protocol3MaxKmipBatchItems	Protocol3MaxKmipBatchItems name
02	Tiny Atom Token	2	Protocol3MaxKmipBatchItems value
F3	End Name Token		End Name-Value pair
F1	End List Token		End HostProperties list
F3	End Name Token		End HostProperties Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00 00	Pad	0	Included in ComPacket and Packet lengths

3.3.3 Common Commands and Responses

The commands defined in this section are commonly used and occur in most communications between a host and a TPer. The commands defined here are referenced in multiple other sections, rather than repeated in each instance.

3.3.3.1 Open Session to Key Per I/O SP as Admin1

This section describes the request message a host sends to a TPer to open a session to the Key Per I/O SP using the Admin1 Authority credential. This section assumes the Admin1 PIN was updated to a new value after Key Per I/O was Activated (see section 3.3.6).

The following pseudo-code is the signature of the `StartSession` method used to open a session to the Key Per I/O SP with the Admin1 authority:

```
session[0:0] -> SMUID.StartSession[HostSessionID : HSN, SPID : KeyPerIOSP_UID, Write
: TRUE, HostChallenge = <Admin1_password>, HostSigningAuthority = Admin1_UID]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the invocation of the `StartSession` method (see Table 5 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 6c 00 00 00 00  00 00 00 00 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 54 00 00 00 00
0030    00 00 00 00 00 00 00 45  f8 a8 00 00 00 00 00 00
0040    00 ff a8 00 00 00 00 00  00 ff 02 f0 01 a8 00 00
0050    02 05 00 00 00 03 01 f2  00 af 41 64 6d 69 6e 31
0060    5f 70 61 73 73 77 6f 72  64 f3 f2 03 a8 00 00 00
0070    09 00 01 00 01 f3 f1 f9  f0 00 00 00 f1 00 00 00
0080    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 5: Open Session to Key Per I/O SP as Admin1

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
`00 00 00 00`	Reserved	0	
`08 00`	ComID	`0x0800`	This field is vendor unique
`00 00`	ComIDExtension	0	This field is vendor unique
`00 00 00 00`	OutstandingData	0	
`00 00 00 00`	MinTransfer	0	
`00 00 00 6C`	Length	108	Number of bytes in Packet
Packet			
`00 00 00 00`	TPerSN	0	TPer Session Number
`00 00 00 00`	HostSN	0	Host Session Number
`00 00 00 00`	SeqNumber	0	
`00 00`	Reserved	0	

Bytes (Hex)	Purpose	Value	Notes
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 54	Length	84	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 45	Length	69	Number of bytes in Payload
Payload			
F8	Call Token		Begin Start Session Method
A8 00 00 00 00 00 00 00 FF	Short Atom Token	SessionManager_UID	Session Manager UID
A8 00 00 00 00 00 00 FF 02	Short Atom Token	StartSession_UID	Start Session Method UID
F0	Start List Token		Begin method parameter list
01	Tiny Atom Token	1	Required Parameter: HostSessionID
A8 00 00 02 05 00 00 00 03	Short Atom Token	KeyPerIO_UID	Required Parameter: KeyPerIOSP UID
01	Tiny Atom Token	1	Required Parameter: Write Session
F2	Start Name Token		Begin Name-Value pair
00	Tiny Atom Token	0	Required Parameter: HostChallenge
AF 41 64 6D 69 6E 31 5F 70 61 73 73 77 6F 72 64	Short Atom Token	Admin1_password	Example password: Admin1_password
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	Required Parameter: HostSigningAuthority
A8 00 00 00 09 00 01 00 01	Short Atom Token	Admin1_UID	Admin1 UID
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list

Bytes (Hex)	Purpose	Value	Notes
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00 00 00	Pad	0	Included in ComPacket and Packet lengths

3.3.3.2 SyncSession Response

This section describes the response a host receives from a TPer following a successful open session request. This response is common to all open session requests in this document where the ComID is `0x0800`, the HostSessionID is `0x1`, and the SPSessionID is `0x1001` (see Table 6).

The following pseudo-code is the signature of the `SyncSession` response to a `StartSession` request:

```
session[0:0] <- SMUID.SyncSession[HostSessionID : 0x00000001, SPSessionID :
0x00001001]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding of the `SyncSession` response from `StartSession` (see Table 6 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 4c 00 00 00 00  00 00 00 00 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 34 00 00 00 00
0030    00 00 00 00 00 00 00 25  f8 a8 00 00 00 00 00 00
0040    00 ff a8 00 00 00 00 00  00 ff 03 f0 84 00 00 00
0050    01 84 00 00 10 01 f1 f9  f0 00 00 00 f1 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 6: SyncSession Response

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00	MinTransfer	0	
00 00 00 4C	Length	76	Number of bytes in Packet
Packet			
00 00 00 00	TPerSN	0	TPer Session Number
00 00 00 00	HostSN	0	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 34	Length	52	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 25	Length	37	Number of bytes in Payload
Payload			
F8	Call Token		Begin Sync Session Method
A8 00 00 00 00 00 00 00 FF	Short Atom Token	SessionManager_UID	Session Manager UID
A8 00 00 00 00 00 00 FF 03	Short Atom Token	SyncSession_UID	Sync Session Method UID
F0	Start List Token		Begin method parameter list
84 00 00 00 01	Short Atom Token	1	Required Parameter: HostSessionID
84 00 00 10 01	Short Atom Token	0x1001	Required Parameter: SPSessionID
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved

Bytes (Hex)	Purpose	Value	Notes
F1	End List Token		End Method Status list
00 00 00	Pad	0	Included in ComPacket and Packet lengths

3.3.3.3 Set Method Response

This section describes the response a host receives from a TPer following a successful `Set` method request. This response is common to all `Set` method requests where the `Set` method is successful.

The following pseudo-code is the signature of the `Set` method response:

```
session[TSN:HSN] <- [ ]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding of `Set` method (see Table 7 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 2c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 14 00 00 00 00
0030    00 00 00 00 00 00 00 08  f0 f1 f9 f0 00 00 00 f1
0040    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 7: Set Method Response

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 2C	Length	44	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	

Bytes (Hex)	Purpose	Value	Notes
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 14	Length	20	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 08	Length	8	Number of bytes in Payload
Payload			
F0	Start List Token		Begin results list
F1	End List Token		End results list
F9	End of Data Token		
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

3.3.3.4 Ending the Session

3.3.3.4.1 Send End of Session Token

This section specifies the request message a host sends to a TPer to close a session.

The following pseudo-code is the signature of the End Of Session token sent to the TPer:

```
session[TSN:HSN] -> EOS
```

Security Protocol 0x01 ComPacket Payload – TCG byte encoding of the ComPacket containing the End Of Session token (see Table 8 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 28 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 10 00 00 00 00
0030    00 00 00 00 00 00 00 01  fa 00 00 00 00 00 00 00
0040    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 8: End Session

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 28	Length	40	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 10	Length	16	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 01	Length	1	Number of bytes in Payload
Payload			
FA	End of Session Token		End the Session
00 00 00	Pad	0	Included in ComPacket and Packet lengths

3.3.3.4.2 End of Session Response

This section specifies the response a host receives from a TPer following an end of session request.
The following pseudo-code is the signature of the TPer's response to the End Of Session token:

```
session[TSN:HSN] <- EOS
```

Security Protocol 0x01 ComPacket Payload – TCG byte encoding of the response to the End Of Session token (see Table 9 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 28 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 10 00 00 00 00
0030    00 00 00 00 00 00 00 01  fa 00 00 00 00 00 00 00
0040    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 9: End Session Response

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 28	Length	40	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 10	Length	16	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 01	Length	1	Number of bytes in Payload
Payload			
FA	End of Session Token		End the Session Response
00 00 00	Pad	0	Included in ComPacket and Packet lengths

3.3.4 Take Ownership of Storage Device

The following subsections describe the procedure a host follows to take ownership of a Storage Device.

3.3.4.1 Open Session to Admin SP as Anybody authority

3.3.4.1.1 StartSession – Admin SP

The following pseudo-code is the signature of the `StartSession` method used to open a session to the Admin SP with the Anybody authority:

```
session[0:0] -> SMUID.StartSession[HostSessionID : 0x01, SPID : AdminSP_UID, Write : True]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `StartSession` method (see Table 10 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 4c 00 00 00 00  00 00 00 00 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 34 00 00 00 00
0030    00 00 00 00 00 00 00 26  f8 a8 00 00 00 00 00 00
0040    00 ff a8 00 00 00 00 00  00 ff 02 f0 01 a8 00 00
0050    02 05 00 00 00 01 01 f1  f9 f0 00 00 00 f1 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
…
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 10: StartSession to Admin SP as Anybody

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
`00 00 00 00`	Reserved	0	
`08 00`	ComID	`0x0800`	This field is vendor unique
`00 00`	ComIDExtension	0	This field is vendor unique
`00 00 00 00`	OutstandingData	0	
`00 00 00 00`	MinTransfer	0	
`00 00 00 4C`	Length	76	Number of bytes in Packet
Packet			
`00 00 00 00`	TPerSN	0	TPer Session Number
`00 00 00 00`	HostSN	0	Host Session Number
`00 00 00 00`	SeqNumber	0	
`00 00`	Reserved	0	
`00 00`	AckType	0	
`00 00 00 00`	Acknowledgement	0	

Bytes (Hex)	Purpose	Value	Notes
00 00 00 34	Length	52	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 26	Length	38	Number of bytes in Payload
Payload			
F8	Call Token		Begin Start Session Method
A8 00 00 00 00 00 00 00 FF	Short Atom Token	SessionManager_UID	Session Manager UID
A8 00 00 00 00 00 00 FF 02	Short Atom Token	StartSession_UID	Start Session Method UID
F0	Start List Token		Begin method parameter list
01	Tiny Atom Token	1	Required Parameter: HostSessionID
A8 00 00 02 05 00 00 00 01	Short Atom Token	AdminSP_UID	Required Parameter: AdminSP UID
01	Tiny Atom Token	1	Required Parameter: Write Session
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00 00	Pad	0	Included in ComPacket and Packet lengths

3.3.4.1.2 SyncSession – Admin SP

The response to the above `StartSession` request is a `SyncSession` response (see section 3.3.3.2).

3.3.4.2 Get MSID PIN Value from C_PIN Table

The following pseudo-code is the signature of the `Get` method on the C_PIN_MSID row in the `C_PIN` Table of the Admin SP:

```
session[TSN:HSN] -> C_PIN_MSID_UID.Get[Cellblock : [startColumn = PIN, endColumn = PIN]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from `Get` method (see Table 11 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 4c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 34 00 00 00 00
0030    00 00 00 00 00 00 00 25  f8 a8 00 00 00 0b 00 00
0040    84 02 a8 00 00 00 06 00  00 00 16 f0 f0 f2 03 03
0050    f3 f2 04 03 f3 f1 f1 f9  f0 00 00 00 f1 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 11: Get MSID PIN Value

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 4C	Length	76	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 34	Length	52	Number of bytes in Subpacket
Subpacket			

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 25	Length	37	Number of bytes in Payload
Payload			
F8	Call Token		Begin Get Method
A8 00 00 00 0B 00 00 84 02	Short Atom Token	C_PIN_MSID_UID	C_PIN_MSID UID
A8 00 00 00 06 00 00 00 16	Short Atom Token	GetMethod_UID	Get Method UID
F0	Start List Token		Begin method parameter list
F0	Start List Token		Begin cell block
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	Start Column
03	Tiny Atom Token	3	PIN Column
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
04	Tiny Atom Token	4	End Column
03	Tiny Atom Token	3	PIN Column
F3	End Name Token		End Name-Value pair
F1	End List Token		End cell block
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00 00 00	Pad	0	Included in ComPacket and Packet lengths

3.3.4.2.1 Response

The following pseudo-code is the signature of the `Get` method response that contains the C_PIN_MSID PIN column value:

```
session[TSN:HSN] <- [ [PIN = <MSID_password>] ]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Get` method response (see Table 12 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 40 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 28 00 00 00 00
0030    00 00 00 00 00 00 00 1b  f0 f0 f2 03 ad 4d 53 49
0040    44 5f 70 61 73 73 77 6f  72 64 f3 f1 f1 f9 f0 00
0050    00 00 f1 00 00 00 00 00  00 00 00 00 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 12: Get MSID PIN Response

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 40	Length	64	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 28	Length	40	Number of bytes in Subpacket
Subpacket			

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 1B	Length	27	Number of bytes in Payload
Payload			
F0	Start List Token		Begin results list
F0	Start List Token		Begin first row of results
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	PIN Column
AD 4D 53 49 44 5F 70 61 73 73 77 6F 72 64	Short Atom Token	MSID_password	Example password: MSID_password
F3	End Name Token		End Name-Value pair
F1	End List Token		End first row of results
F1	End List Token		End results list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00	Pad	0	Included in ComPacket and Packet lengths

3.3.4.3 Close Session

The open Session to the Admin SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.4.4 Open Session to Admin SP as SID Authority using <MSID_password>

The following pseudo-code is the signature of the `StartSession` method used to open a session to the Admin SP with SID authority:

```
session[0:0] -> SMUID.StartSession[HostSessionID : HSN, SPID : AdminSP_UID, Write :
True, HostChallenge = <MSID_password>, HostSigningAuthority = SID_UID]
```

Security Protocol 0x01 ComPacket Payload – TCG Method byte encoding from the invocation of the StartSession method (see Table 13 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 68 00 00 00 00  00 00 00 00 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 50 00 00 00 00
0030    00 00 00 00 00 00 00 43  f8 a8 00 00 00 00 00 00
0040    00 ff a8 00 00 00 00 00  00 ff 02 f0 01 a8 00 00
0050    02 05 00 00 00 01 01 f2  00 ad 4d 53 49 44 5f 70
0060    61 73 73 77 6f 72 64 f3  f2 03 a8 00 00 00 09 00
0070    00 00 06 f3 f1 f9 f0 00  00 00 f1 00 00 00 00 00
0080    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 13: Open Session to Admin SP as SID

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 68	Length	104	Number of bytes in Packet
Packet			
00 00 00 00	TPerSN	0	TPer Session Number
00 00 00 00	HostSN	0	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 50	Length	80	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 43	Length	67	Number of bytes in Payload

Bytes (Hex)	Purpose	Value	Notes
Payload			
F8	Call Token		Begin Start Session Method
A8 00 00 00 00 00 00 00 FF	Short Atom Token	SessionManager_UID	Session Manager UID
A8 00 00 00 00 00 00 FF 02	Short Atom Token	StartSession_UID	Start Session Method UID
F0	Start List Token		Begin method parameter list
01	Tiny Atom Token	1	Required Parameter: HostSessionID
A8 00 00 02 05 00 00 00 01	Short Atom Token	AdminSP_UID	Required Parameter: AdminSP UID
01	Tiny Atom Token	1	Required Parameter: Write Session
F2	Start Name Token		Begin Name-Value pair
00	Tiny Atom Token	0	Required Parameter: HostChallenge
AD 4D 53 49 44 5F 70 61 73 73 77 6F 72 64	Short Atom Token	MSID_password	Example password: MSID_password
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	Required Parameter: HostSigningAuthority
A8 00 00 00 09 00 00 00 06	Short Atom Token	SID_UID	SID UID
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

Bytes (Hex)	Purpose	Value	Notes
00	Pad	0	Included in ComPacket and Packet lengths

The response to the above `StartSession` request is a `SyncSession` response (see section 3.3.3.2).

3.3.4.5 Set <new_SID_password> Value in SID C_PIN credential PIN column

The following pseudo-code is the signature of the `Set` method on the C_PIN_SID PIN column of the Admin SP to change the SID PIN:

```
session[TSN:HSN] -> C_PIN_SID_UID.Set[Values = [PIN = <new_SID_password>]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from `Set` method (see Table 14 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 5c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 44 00 00 00 00
0030    00 00 00 00 00 00 00 35  f8 a8 00 00 00 0b 00 00
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    03 d0 10 6e 65 77 5f 53  49 44 5f 70 61 73 73 77
0060    6f 72 64 f3 f1 f3 f1 f9  f0 00 00 00 f1 00 00 00
0070    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 14: Change SID PIN

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 5C	Length	92	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	

Bytes (Hex)	Purpose	Value	Notes
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 44	Length	68	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 35	Length	53	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 00 0B 00 00 00 01	Short Atom Token	C_PIN_SID_UID	C_PIN_SID UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	PIN Column
D0 10 6E 65 77 5F 53 49 44 5F 70 61 73 73 77 6F 72 64	Medium Atom Token	new_SID_password	Example new password: new_SID_password
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved

Bytes (Hex)	Purpose	Value	Notes
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00 00 00	Pad	0	Included in ComPacket and Packet lengths

The response to the above `Set` method request is a `Set` method response (see section 3.3.3.3).

3.3.4.6 Close Session

The open Session to the Admin SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.5 Activate Key Per I/O SP

The following subsections describe the procedure a host follows to Activate the Key Per I/O SP.

3.3.5.1 Open Session to Admin SP as SID Authority

The following pseudo-code is the signature of the `StartSession` method used to open a session to the Admin SP with SID authority:

```
session[0:0] -> SMUID.StartSession[HostSessionID : HSN, SPID : AdminSP_UID, Write :
TRUE, HostChallenge = new_SID_password, HostSigningAuthority = SID_UID]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `StartSession` method (see Table 15 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 6c 00 00 00 00  00 00 00 00 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 54 00 00 00 00
0030    00 00 00 00 00 00 00 47  f8 a8 00 00 00 00 00 00
0040    00 ff a8 00 00 00 00 00  00 ff 02 f0 01 a8 00 00
0050    02 05 00 00 00 01 01 f2  00 d0 10 6e 65 77 5f 53
0060    49 44 5f 70 61 73 73 77  6f 72 64 f3 f2 03 a8 00
0070    00 00 09 00 00 00 06 f3  f1 f9 f0 00 00 00 f1 00
0080    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 15: Open Session to Admin SP as SID

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique

Bytes (Hex)	Purpose	Value	Notes
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 6C	Length	108	Number of bytes in Packet
Packet			
00 00 00 00	TPerSN	0	TPer Session Number
00 00 00 00	HostSN	0	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 54	Length	84	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 47	Length	71	Number of bytes in Payload
Payload			
F8	Call Token		Begin Start Session Method
A8 00 00 00 00 00 00 00 FF	Short Atom Token	SessionManager_UID	Session Manager UID
A8 00 00 00 00 00 00 FF 02	Short Atom Token	StartSession_UID	Start Session Method UID
F0	Start List Token		Begin method parameter list
01	Tiny Atom Token	1	Required Parameter: HostSessionID
A8 00 00 02 05 00 00 00 01	Short Atom Token	AdminSP_UID	Required Parameter: AdminSP UID
01	Tiny Atom Token	1	Required Parameter: Write Session
F2	Start Name Token		Begin Name-Value pair

Bytes (Hex)	Purpose	Value	Notes
00	Tiny Atom Token	0	Required Parameter: HostChallenge
D0 10 6E 65 77 5F 53 49 44 5F 70 61 73 73 77 6F 72 64	Medium Atom Token	new_SID_password	Example password: new_SID_password
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	Required Parameter: HostSigningAuthority
A8 00 00 00 09 00 00 00 06	Short Atom Token	SID_UID	SID UID
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00	Pad	0	Included in ComPacket and Packet lengths

The response to the above `StartSession` request is a `SyncSession` response (see section 3.3.3.2).

3.3.5.2 Determine Life Cycle State of Key Per I/O SP

The following pseudo-code is the signature of the `Get` method used to retrieve the LifeCycle column of the Key Per I/O SP:

```
session[TSN:HSN] -> KeyPerIOSP_UID.Get[Cellblock : [startColumn = LifeCycle,
endColumn = LifeCycle]]
```

Security Protocol 0x01 ComPacket Payload – TCG Method byte encoding from the `Get` method (see Table 16 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 4c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 34 00 00 00 00
0030    00 00 00 00 00 00 00 25  f8 a8 00 00 02 05 00 00
0040    00 03 a8 00 00 00 06 00  00 00 16 f0 f0 f2 03 06
0050    f3 f2 04 06 f3 f1 f1 f9  f0 00 00 00 f1 00 00 00
```

```
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 16: Get Life Cycle State of Key Per I/O SP

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 4C	Length	76	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 34	Length	52	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 25	Length	37	Number of bytes in Payload
Payload			
F8	Call Token		Begin Get Method
A8 00 00 02 05 00 00 00 03	Short Atom Token	KeyPerIO_UID	KeyPerIOSP UID
A8 00 00 00 06 00 00 00 16	Short Atom Token	GetMethod_UID	Get Method UID
F0	Start List Token		Begin method parameter list
F0	Start List Token		Begin cell block

Bytes (Hex)	Purpose	Value	Notes
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	Start Column
06	Tiny Atom Token	6	LifeCycle Column
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
04	Tiny Atom Token	4	End Column
06	Tiny Atom Token	6	LifeCycle Column
F3	End Name Token		End Name-Value pair
F1	End List Token		End cell block
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00 00 00	Pad	0	Included in ComPacket and Packet lengths

3.3.5.2.1 Response

The following pseudo-code is the signature of the `Get` method response of the LifeCycle column for the Key Per I/O SP that is currently Manufactured-Inactive:

```
session[TSN:HSN] <- [ [LifeCycle = Manufactured-Inactive] ]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Get` method response (see Table 17 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 34 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 1c 00 00 00 00
0030    00 00 00 00 00 00 00 0e  f0 f0 f2 06 08 f3 f1 f1
0040    f9 f0 00 00 00 f1 00 00  00 00 00 00 00 00 00 00
0050    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 17: Get Life Cycle State Response

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 34	Length	52	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 1C	Length	28	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 0E	Length	14	Number of bytes in Payload
Payload			
F0	Start List Token		Begin results list
F0	Start List Token		Begin first row of results
F2	Start Name Token		Begin Name-Value pair
06	Tiny Atom Token	6	LifeCycle Column
08	Tiny Atom Token	8	Manufactured-Inactive value
F3	End Name Token		End Name-Value pair
F1	End List Token		End first row of results
F1	End List Token		End results list

Bytes (Hex)	Purpose	Value	Notes
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00 00	Pad	0	Included in ComPacket and Packet lengths

3.3.5.3 Activate Key Per I/O SP Using Activate Method on Key Per I/O SP object in Admin SP

Note: If the Key Per I/O SP is successfully activated, the PIN for Admin1 in the Key Per I/O SP is set to the current SID PIN, which is “new_SID_password”.

The following pseudo-code is the signature of the `Activate` method on the Key Per I/O SP:

```
session[TSN:HSN] => KeyPerIOSP_UID.Activate[]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Activate` method (see Table 18 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 40 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 28 00 00 00 00
0030    00 00 00 00 00 00 00 1b  f8 a8 00 00 02 05 00 00
0040    00 03 a8 00 00 00 06 00  00 02 03 f0 f1 f9 f0 00
0050    00 00 f1 00 00 00 00 00  00 00 00 00 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 18: Activate Key Per I/O SP

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 40	Length	64	Number of bytes in Packet
Packet			

Bytes (Hex)	Purpose	Value	Notes
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 28	Length	40	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 1B	Length	27	Number of bytes in Payload
Payload			
F8	Call Token		Begin Activate Method
A8 00 00 02 05 00 00 00 03	Short Atom Token	KeyPerIO_UID	Invoking UID
A8 00 00 00 06 00 00 02 03	Short Atom Token	Activate_UID	Method UID
F0	Start List Token		Begin method parameter list
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00	Pad	0	Included in ComPacket and Packet lengths

3.3.5.3.1 Activate Method Response

The following pseudo-code is the signature of the `Activate` method response:

```
session[TSN:HSN] <- []
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Activate` method response (see Table 19 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 2c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 14 00 00 00 00
0030    00 00 00 00 00 00 00 08  f0 f1 f9 f0 00 00 00 f1
0040    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 19: Activate Key Per I/O SP Response

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 2C	Length	44	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 14	Length	20	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 08	Length	8	Number of bytes in Payload
Payload			
F0	Start List Token		Begin results list

Bytes (Hex)	Purpose	Value	Notes
F1	End List Token		End results list
F9	End of Data Token		
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

3.3.5.4 Close Session

The open Session to the Admin SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.6 Change Admin1 PIN in Key Per I/O SP

The following subsections describe the procedure a host follows to change the PIN associated with the Admin1 authority in the Key Per I/O SP.

3.3.6.1 Open Session to Key Per I/O SP as Admin1

Note: When the Key Per I/O SP was activated, the current SID PIN (C_PIN_SID) was copied into the PIN column of the Admin1 PIN (C_PIN_Admin1).

The following pseudo-code is the signature of the `StartSession` method used to open a session to the Key Per I/O SP with Admin1 authority:

```
session[0:0] -> SMUID.StartSession[HostSessionID : HSN, SPID : KeyPerIOSP_UID, Write
: TRUE, HostChallenge = <new_SID_password>, HostSigningAuthority = Admin1_UID]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `StartSession` method (see Table 20 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 6c 00 00 00 00  00 00 00 00 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 54 00 00 00 00
0030    00 00 00 00 00 00 00 47  f8 a8 00 00 00 00 00 00
0040    00 ff a8 00 00 00 00 00  00 ff 02 f0 01 a8 00 00
0050    02 05 00 00 00 03 01 f2  00 d0 10 6e 65 77 5f 53
0060    49 44 5f 70 61 73 73 77  6f 72 64 f3 f2 03 a8 00
0070    00 00 09 00 01 00 01 f3  f1 f9 f0 00 00 00 f1 00
0080    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 20: Open Session to Key Per I/O SP as Admin1

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 6C	Length	108	Number of bytes in Packet
Packet			
00 00 00 00	TPerSN	0	TPer Session Number
00 00 00 00	HostSN	0	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 54	Length	84	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 47	Length	71	Number of bytes in Payload
Payload			
F8	Call Token		Begin Start Session Method
A8 00 00 00 00 00 00 00 FF	Short Atom Token	SessionManager_UID	Session Manager UID
A8 00 00 00 00 00 00 FF 02	Short Atom Token	StartSession_UID	Start Session Method UID
F0	Start List Token		Begin method parameter list
01	Tiny Atom Token	1	Required Parameter: HostSessionID

Bytes (Hex)	Purpose	Value	Notes
A8 00 00 02 05 00 00 00 03	Short Atom Token	KeyPerIO_UID	Required Parameter: KeyPerIOSP UID
01	Tiny Atom Token	1	Required Parameter: Write Session
F2	Start Name Token		Begin Name-Value pair
00	Tiny Atom Token	0	Required Parameter: HostChallenge
D0 10 6E 65 77 5F 53 49 44 5F 70 61 73 73 77 6F 72 64	Medium Atom Token	new_SID_password	Example password: new_SID_password
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	Required Parameter: HostSigningAuthority
A8 00 00 00 09 00 01 00 01	Short Atom Token	Admin1_UID	Admin1 UID
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00	Pad	0	Included in ComPacket and Packet lengths

The response to the above `StartSession` request is a `SyncSession` response (see section 3.3.3.2).

3.3.6.2 Set <Admin1_password> Value in Admin1 C_PIN Credential PIN column

The following pseudo-code is the signature of the `Set` method used to change the C_PIN_Admin1 PIN of the Key Per I/O SP to <Admin1_password>:

```
session[TSN:HSN] -> C_PIN_Admin1_UID.Set[Values = [PIN = <Admin1_password>]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from `Set` method (see Table 21 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 5c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 44 00 00 00 00
0030    00 00 00 00 00 00 00 38  f8 a8 00 00 00 0b 00 01
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    03 d0 13 6e 65 77 5f 41  64 6d 69 6e 31 5f 70 61
0060    73 73 77 6f 72 64 f3 f1  f3 f1 f9 f0 00 00 00 f1
0070    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 21: Change Admin1 PIN

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 5C	Length	92	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 44	Length	68	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 38	Length	56	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method

A8 00 00 00 0B 00 01 00 01	Short Atom Token	C_PIN_Admin1_UID	C_PIN_Admin1 UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	PIN Column
D0 13 6E 65 77 5F 41 64 6D 69 6E 31 5F 70 61 73 73 77 6F 72 64	Medium Atom Token	new_Admin1_password	Example new password: new_Admin1_password
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

The response to the above `Set` method request is a `Set` method response (see section 3.3.3.3).

3.3.6.3 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.7 Configure Namespace to be Managed by Key Per I/O

The following subsections describe the procedure a host follows to change the configuration of a Namespace so the Namespace is managed by the Key Per I/O SP [3].

3.3.7.1 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.3.7.2 Set Managed Column to TRUE for Namespace

The following pseudo-code is the signature of the `Set` method used to change the Managed column of the KeyTagAllocation1 row to TRUE:

```
session[TSN:HSN] -> KeyTagAllocation1_UID.Set[Values = [Managed = TRUE]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from `Set` method (see Table 22 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 48 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 30 00 00 00 00
0030    00 00 00 00 00 00 00 24  f8 a8 00 00 12 01 00 00
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    04 01 f3 f1 f3 f1 f9 f0  00 00 00 f1 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 22: Set Namespace Managed column to TRUE

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 48	Length	72	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	

Bytes (Hex)	Purpose	Value	Notes
00 00 00 30	Length	48	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 24	Length	36	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 01 00 00 00 01	Short Atom Token	Key_Tag_Allocation1_UID	KeyTagAllocation1 UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
04	Tiny Atom Token	4	Managed Column
01	Tiny Atom Token	1	true
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

The response to the above `Set` method request is a `Set` method response (see section 3.3.3.3).

3.3.7.3 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.8 Enable Key Injection Interface

The following subsections describe the procedure a host follows to enable the Key Injection Interface of a TPer.

3.3.8.1 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.3.8.2 Unlock Key Injection Interface by setting KeyInjectionInterfaceLocked column to FALSE

The following pseudo-code is the signature of the `Set` method to change the KeyInjectionInterfaceLocked column of the `KPIOPolicies` single row to FALSE:

```
session[TSN:HSN] -> KPIOPolicies_SingleRow_UID.Set[Values =
[KeyInjectionInterfaceLocked = FALSE]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Set` method (see Table 23 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 48 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 30 00 00 00 00
0030    00 00 00 00 00 00 00 24  f8 a8 00 00 12 03 00 00
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    07 00 f3 f1 f3 f1 f9 f0  00 00 00 f1 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 23: Enable Key Injection Interface

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
`00 00 00 00`	Reserved	0	
`08 00`	ComID	`0x0800`	This field is vendor unique
`00 00`	ComIDExtension	0	This field is vendor unique
`00 00 00 00`	OutstandingData	0	
`00 00 00 00`	MinTransfer	0	

Bytes (Hex)	Purpose	Value	Notes
00 00 00 48	Length	72	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 30	Length	48	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 24	Length	36	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 03 00 00 00 01	Short Atom Token	KPIO_Policies_UID	KPIOPolicies Row UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
07	Tiny Atom Token	7	KeyInjectionInterfaceLocked Column
00	Tiny Atom Token	0	false
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair

Bytes (Hex)	Purpose	Value	Notes
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

The response to the above `Set` method request is a `Set` method response (see section 3.3.3.3).

3.3.8.3 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.9 Unlock Access to KEK

The following subsections describe the procedure a host follows to change the KEK access state to Access Unlocked [3].

3.3.9.1 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.3.9.2 Unlock Access to KEK by setting the AccessLocked column in the KeyEncryptionKey Table to FALSE

The following pseudo-code is the signature of the `Set` method that changes the AccessLocked column of the KeyEncryptionKey1 row to FALSE:

```
session[TSN:HSN] -> KeyEncryptionKey1_UID.Set[Values = [AccessLocked = FALSE]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Set` method (see Table 24 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 48 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 30 00 00 00 00
0030    00 00 00 00 00 00 00 24  f8 a8 00 00 12 02 00 01
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    04 00 f3 f1 f3 f1 f9 f0  00 00 00 f1 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
```

```
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 24: Unlock Access to KEK

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 48	Length	72	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 30	Length	48	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 24	Length	36	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 02 00 01 00 01	Short Atom Token	Key_Encryption_Key1_UID	KeyEncryptionKey1 UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list

Bytes (Hex)	Purpose	Value	Notes
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
04	Tiny Atom Token	4	AccessLocked Column
00	Tiny Atom Token	0	false
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

The response to the above `Set` method request is a `Set` method response (see section 3.3.3.3).

3.3.9.3 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.10 Associate KEK with Namespace

The following subsections describe the procedure a host follows to configure a Namespace in a TPer, to associate the Namespace with a KEK.

3.3.10.1 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.3.10.2 Configure AllowedKeyEncryptionKeys to associate KEK with Namespace

The following pseudo-code is the signature of the `Set` method used to change the AllowedKeyEncryptionKeys column of the KeyTagAllocation1 row so that the column contains the KeyEncryptionKey1 UID:

```
session[TSN:HSN] -> KeyTagAllocation1_UID.Set[Values = [AllowedKeyEncryptionKeys
=[KeyEncryptionKey1_UID]]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Set` method (see Table 25 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 54 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 3c 00 00 00 00
0030    00 00 00 00 00 00 00 2e  f8 a8 00 00 12 01 00 00
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    06 f0 a8 00 00 12 02 00  01 00 01 f1 f3 f1 f3 f1
0060    f9 f0 00 00 00 f1 00 00  00 00 00 00 00 00 00 00
0070    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 25: Associate KEK with Namespace

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 54	Length	84	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 3C	Length	60	Number of bytes in Subpacket
Subpacket			

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 2E	Length	46	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 01 00 00 00 01	Short Atom Token	Key_Tag_Allocation1_UID	KeyTagAllocation1 UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
06	Tiny Atom Token	6	AllowedKeyEncryptionKeys Column
F0	Start List Token		Begin kek_obj_uidref_list list
A8 00 00 12 02 00 01 00 01	Short Atom Token	Key_Encryption_Key1_UID	KeyEncryptionKey1 UID
F1	End List Token		End kek_obj_uidref_list list
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

Bytes (Hex)	Purpose	Value	Notes
00 00	Pad	0	Included in ComPacket and Packet lengths

3.3.10.3 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.11 Enable Plaintext KEK Provisioning for all KEKs

The following subsections describe the procedure a host follows to change the KPIO Policies configuration in a TPer to enable plaintext KEK provisioning for all KEKs.

3.3.11.1 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.3.11.2 Configure PlaintextKEKProgrammingEnabled column to TRUE

The following pseudo-code is the signature of the `Set` method used to change the PlaintextKEKProgrammingEnabled column of the `KPIOPolicies` single row to TRUE:

```
session[TSN:HSN] -> KPIOPolicies_SingleRow_UID.Set[Values =
[PlaintextKEKProgrammingEnabled = TRUE]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Set` method (see Table 26 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 48 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 30 00 00 00 00
0030    00 00 00 00 00 00 00 24  f8 a8 00 00 12 03 00 00
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    05 01 f3 f1 f3 f1 f9 f0  00 00 00 f1 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 26: Enable Plaintext KEK Provisioning for all KEKS

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 48	Length	72	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 30	Length	48	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 24	Length	36	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 03 00 00 00 01	Short Atom Token	KPIO_Policies_UID	KPIOPolicies Row UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
05	Tiny Atom Token	5	PlaintextKEKProgrammingEnabled Column
01	Tiny Atom Token	1	true
F3	End Name Token		End Name-Value pair

Bytes (Hex)	Purpose	Value	Notes
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

The response to the above `Set` method request is a `Set` method response (see section 3.3.3.3).

3.3.11.3 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.12 Enable Plaintext KEK Provisioning for Single KEK

The following subsections describe the procedure a host follows to enable plaintext KEK provisioning for an individual KEK in a TPer.

3.3.12.1 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.3.12.2 Configure AllowedKeyEncryptionKeys to include NULLKeyEncryptionKey UID

The following pseudo-code is the signature of the `Set` method that changes the AllowedKeyEncryptionKeys column of the KeyEncryptionKey1 row that contains the values NULLKeyEncryptionKey UID and KeyEncryptionKey1 UID:

```
session[TSN:HSN] -> KeyEncryptionKey1_UID.Set[Values = [AllowedKeyEncryptionKeys =
[NULLKeyEncryptionKey_UID, KeyEncryptionKey1_UID]]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Set` method (see Table 27 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 5c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 44 00 00 00 00
0030    00 00 00 00 00 00 00 37  f8 a8 00 00 12 02 00 01
```

```
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    06 f0 a8 00 00 12 02 00  00 00 01 a8 00 00 12 02
0060    00 01 00 01 f1 f3 f1 f3  f1 f9 f0 00 00 00 f1 00
0070    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 27: Enable Plaintext KEK Provisioning for Single KEK

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 5C	Length	92	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 44	Length	68	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 37	Length	55	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 02 00 01 00 01	Short Atom Token	Key_Encryption_Key1_UID	KeyEncryptionKey1 UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID

Bytes (Hex)	Purpose	Value	Notes
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
06	Tiny Atom Token	6	AllowedKeyEncryptionKeys Column
F0	Start List Token		Begin kek_obj_uidref_list list
A8 00 00 12 02 00 00 00 01	Short Atom Token	NULLKeyEncryptionKey_UID	NULLKeyEncryptionKey UID
A8 00 00 12 02 00 01 00 01	Short Atom Token	Key_Encryption_Key1_UID	KeyEncryptionKey1 UID
F1	End List Token		End kek_obj_uidref_list list
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00	Pad	0	Included in ComPacket and Packet lengths

The response to the above `Set` method request is a `Set` method response (see section 3.3.3.3).

3.3.12.3 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.13 Enable KEK wrapping with Public Key for all KEKs

The following subsections describe the procedure a host follows to change the KPIO Policies configuration in a TPer to enable injection of KEKs wrapped with a public key, where the policy applies to all KEKs.

3.3.13.1 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.3.13.2 Configure PKIProtectedKEKProgrammingEnabled Column to TRUE

The following pseudo-code is the signature of the `Set` method used to change the PKIProtectedKEKProgrammingEnabled column of the `KPIOPolicies` single row to TRUE:

```
session[TSN:HSN] -> KPIOPolicies_SingleRow_UID.Set[Values =
[PKIProtectedKEKProgrammingEnabled = TRUE]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Set` method (see Table 28 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 48 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 30 00 00 00 00
0030    00 00 00 00 00 00 00 24  f8 a8 00 00 12 03 00 00
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    04 01 f3 f1 f3 f1 f9 f0  00 00 00 f1 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 28: Enable KEK Wrapping with Public Key for all KEKS

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
`00 00 00 00`	Reserved	0	
`08 00`	ComID	`0x0800`	This field is vendor unique
`00 00`	ComIDExtension	0	This field is vendor unique
`00 00 00 00`	OutstandingData	0	
`00 00 00 00`	MinTransfer	0	
`00 00 00 48`	Length	72	Number of bytes in Packet
Packet			
`00 00 10 01`	TPerSN	`0x1001`	TPer Session Number
`00 00 00 01`	HostSN	1	Host Session Number

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 30	Length	48	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 24	Length	36	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 03 00 00 00 01	Short Atom Token	KPIO_Policies_UID	KPIOPolicies Row UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
04	Tiny Atom Token	4	PKIProtectedKEKProgrammingEnabled Column
01	Tiny Atom Token	1	true
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list

Bytes (Hex)	Purpose	Value	Notes
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

The response to the above `Set` method request is a `Set` method response (see section 3.3.3.3).

3.3.13.3 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.14 Enable KEK wrapping with Public Key for Single KEK

The following subsections describe the procedure a host follows to enable injection into a TPer of KEKs wrapped with a public key, where the policy applies to an individual KEK.

3.3.14.1 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.3.14.2 Configure AllowedKeyEncryptionKeys to include PKIPublicKeyEncryptionKey UID

The following pseudo-code is the signature of the `Set` method used to change the AllowedKeyEncryptionKeys column of the KeyEncryptionKey1 row that contains the values PKIPublicKeyEncryptionKey UID and KeyEncryptionKey1 UID:

```
session[TSN:HSN] -> KeyEncryptionKey1_UID.Set[Values = [AllowedKeyEncryptionKeys =
[PKIPublicKeyEncryptionKey_UID, KeyEncryptionKey1_UID]]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Set` method (see Table 29 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 5c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 44 00 00 00 00
0030    00 00 00 00 00 00 00 37  f8 a8 00 00 12 02 00 01
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    06 f0 a8 00 00 12 02 00  00 00 02 a8 00 00 12 02
0060    00 01 00 01 f1 f3 f1 f3  f1 f9 f0 00 00 00 f1 00
0070    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 29: Enable KEK Wrapping with Public Key for Single KEK

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 5C	Length	92	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 44	Length	68	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 37	Length	55	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 02 00 01 00 01	Short Atom Token	Key_Encryption_Key1_UID	KeyEncryptionKey1 UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair

Bytes (Hex)	Purpose	Value	Notes
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
06	Tiny Atom Token	6	AllowedKeyEncryptionKeys Column
F0	Start List Token		Begin kek_obj_uidref_list list
A8 00 00 12 02 00 00 00 02	Short Atom Token	PKIPublicKeyEncryptionKey_UID	PKIPublicKeyEncryptionKey UID
A8 00 00 12 02 00 01 00 01	Short Atom Token	Key_Encryption_Key1_UID	KeyEncryptionKey1 UID
F1	End List Token		End kek_obj_uidref_list list
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00	Pad	0	Included in ComPacket and Packet lengths

The response to the above `Set` method request is a `Set` method response (see section 3.3.3.3).

3.3.14.3 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.15 Clear Single MEK from TPer Key Cache

The following subsections describe the procedure a host follows to clear a single MEK from the TPer's key cache.

3.3.15.1 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.3.15.2 Enable Clear Single MEK Command

The following pseudo-code is the signature of the `Set` method used to change the ClearSingleMEKAllowed column of the `KPIOPolicies` single row to TRUE:

```
session[TSN:HSN] -> KPIOPolicies_SingleRow_UID.Set[Values = [ClearSingleMEKAllowed = TRUE]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Set` method (see Table 30 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 48 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 30 00 00 00 00
0030    00 00 00 00 00 00 00 24  f8 a8 00 00 12 03 00 00
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    01 01 f3 f1 f3 f1 f9 f0  00 00 00 f1 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 30: Enable Clear Single MEK Command

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
`00 00 00 00`	Reserved	0	
`08 00`	ComID	`0x0800`	This field is vendor unique
`00 00`	ComIDExtension	0	This field is vendor unique
`00 00 00 00`	OutstandingData	0	
`00 00 00 00`	MinTransfer	0	
`00 00 00 48`	Length	72	Number of bytes in Packet
Packet			
`00 00 10 01`	TPerSN	`0x1001`	TPer Session Number
`00 00 00 01`	HostSN	1	Host Session Number
`00 00 00 00`	SeqNumber	0	
`00 00`	Reserved	0	

Bytes (Hex)	Purpose	Value	Notes
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 30	Length	48	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 24	Length	36	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 03 00 00 00 01	Short Atom Token	KPIO_Policies_UID	KPIOPolicies Row UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	ClearSingleMEKAllowed Column
01	Tiny Atom Token	1	true
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved

Bytes (Hex)	Purpose	Value	Notes
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

3.3.15.3 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.15.4 Clear Single MEK Request

Security Protocol 0x02 – Clear Single MEK Command Request:

```
IF_SEND(ProtocolID=2, ComID=0x0800, NSID=0x1, KeyTag=0x1)
```

Table 31: Clear Single MEK Command

Byte	Value (Hex)	Description
0 to 3	00 00 08 00	Extended ComID
4 to 7	00 00 00 03	Clear Single MEK Request Code
8 to 9	00 01	Key Tag
10 to 11	00 04	four Bytes of Data
10 to TRNSFLEN-1	00 … 00	Reserved

3.3.15.5 Clear Single MEK Response

Security Protocol 0x02 – Clear Single MEK Command Response:

```
IF_RECV(ProtocolID=2, ComID=0x0800, NSID=0x1)
```

Table 32: Clear Single MEK Response

Byte	Value (Hex)	Description
0 to 3	00 00 08 00	Extended ComID
4 to 7	00 00 00 03	Clear Single MEK Request Code
8 to 9	00 00	Reserved
10 to 11	00 04	four Bytes of Data
12 to 15	00 00 00 00	Success Status
16 to TRNSFLEN-1	00 … 00	Reserved

3.3.16 Clear All MEKs for Namespace from TPer Key Cache

The following subsections describe the procedure a host follows to clear all MEKs from the TPer's key cache, where the MEKs are associated with a specified Namespace.

3.3.16.1 Open Session to Key Per I/O as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.3.16.2 Enable Clear All MEKs Command

The following pseudo-code is the signature of the `Set` method used to change the ClearAllMEKsAllowed column of the `KPIOPolicies` single row to TRUE:

```
session[TSN:HSN] -> KPIOPolicies_SingleRow_UID.Set[Values = [ClearAllMEKsAllowed = TRUE]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Set` method (see Table 33 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 48 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 30 00 00 00 00
0030    00 00 00 00 00 00 00 24  f8 a8 00 00 12 03 00 00
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    02 01 f3 f1 f3 f1 f9 f0  00 00 00 f1 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 33: Enable Clear All MEKs Command

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
`00 00 00 00`	Reserved	0	
`08 00`	ComID	`0x0800`	This field is vendor unique
`00 00`	ComIDExtension	0	This field is vendor unique
`00 00 00 00`	OutstandingData	0	
`00 00 00 00`	MinTransfer	0	
`00 00 00 48`	Length	72	Number of bytes in Packet
Packet			
`00 00 10 01`	TPerSN	`0x1001`	TPer Session Number
`00 00 00 01`	HostSN	1	Host Session Number
`00 00 00 00`	SeqNumber	0	
`00 00`	Reserved	0	
`00 00`	AckType	0	
`00 00 00 00`	Acknowledgement	0	
`00 00 00 30`	Length	48	Number of bytes in Subpacket
Subpacket			

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 24	Length	36	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 03 00 00 00 01	Short Atom Token	KPIO_Policies_UID	KPIOPolicies Row UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
02	Tiny Atom Token	2	ClearAllMEKsAllowed Column
01	Tiny Atom Token	1	true
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

3.3.16.3 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.3.16.4 Clear All MEKs Request

Security Protocol `0x02` – Clear All MEKs Command Request:

```
IF_SEND(ProtocolID=2, ComID=0x0800, NSID=0x1)
```

Table 34: Clear All MEKs Command

Byte	Value (Hex)	Description
0 to 3	`00 00 08 00`	Extended ComID
4 to 7	`00 00 00 04`	Clear All MEKs Request Code
8 to TRNSFLEN-1	`00 … 00`	Reserved

3.3.16.5 Clear All MEKs Response

Security Protocol `0x02` – Clear All MEKs Command Response:

```
IF_RECV(ProtocolID=2, ComID=0x0800, NSID=0x1)
```

Table 35: Clear All MEKs Response

Byte	Value (Hex)	Description
0 to 3	`00 00 08 00`	Extended ComID
4 to 7	`00 00 00 04`	Clear All MEKs Request Code
8 to 9	`00 00`	Reserved
10 to 11	`00 04`	four Bytes of Data
12 to 15	`00 00 00 00`	Success Status
16 to TRNSFLEN-1	`00 … 00`	Reserved

3.3.17 Revert Key Per I/O SP

The following subsections describe the procedure a host follows to Revert the Key Per I/O SP.

3.3.17.1 Open Session to Admin SP as SID

The procedure a host follows to open a Session to the Admin SP as the SID authority is described in section 3.3.5.1.

3.3.17.2 Revert Key Per I/O SP

The following pseudo-code is the signature of the `Revert` method of the Key Per I/O SP:

```
session[TSN:HSN] -> KeyPerIO_UID.Revert[]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Revert` method (see Table 36 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 40 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 28 00 00 00 00
0030    00 00 00 00 00 00 00 1b  f8 a8 00 00 02 05 00 00
```

```
0040    00 03 a8 00 00 00 06 00  00 02 02 f0 f1 f9 f0 00
0050    00 00 f1 00 00 00 00 00  00 00 00 00 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 36: Revert Key Per I/O SP

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 40	Length	64	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 28	Length	40	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 1B	Length	27	Number of bytes in Payload
Payload			
F8	Call Token		Begin Revert Method
A8 00 00 02 05 00 00 00 03	Short Atom Token	KeyPerIO_UID	Invoking UID
A8 00 00 00 06 00 00 02 02	Short Atom Token	Revert_UID	Method UID
F0	Start List Token		Begin method parameter list

Bytes (Hex)	Purpose	Value	Notes
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00	Pad	0	Included in ComPacket and Packet lengths

3.3.17.2.1 Revert Key Per I/O SP Response

The following pseudo-code is the signature of the response to the `Revert` method of the Key Per I/O SP:

```
session[TSN:HSN] <- []
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Revert` method response (see Table 37 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 2c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 14 00 00 00 00
0030    00 00 00 00 00 00 00 08  f0 f1 f9 f0 00 00 00 f1
0040    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 37: Revert Key Per I/O SP Response

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 2C	Length	44	Number of bytes in Packet
Packet			

Bytes (Hex)	Purpose	Value	Notes
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 14	Length	20	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 08	Length	8	Number of bytes in Payload
Payload			
F0	Start List Token		Begin results list
F1	End List Token		End results list
F9	End of Data Token		
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

Note: A TPer aborts the session immediately after reporting the `Revert` result.

3.3.18 Revert TPer

The following subsections describe the procedure a host follows to Revert the Admin SP.

3.3.18.1 Open Session to Admin SP as SID

The procedure a host follows to open a Session to the Admin SP as the SID authority is described in section 3.3.5.1.

3.3.18.2 Revert TPer

The following pseudo-code is the signature of the `Revert` method of the Admin SP:

```
session[TSN:HSN] -> AdminSP_UID.Revert[]
```

Security Protocol 0x01 ComPacket Payload – TCG Method byte encoding from the Revert method (see Table 38 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 40 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 28 00 00 00 00
0030    00 00 00 00 00 00 00 1b  f8 a8 00 00 02 05 00 00
0040    00 01 a8 00 00 00 06 00  00 02 02 f0 f1 f9 f0 00
0050    00 00 f1 00 00 00 00 00  00 00 00 00 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 38: Revert TPer

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 40	Length	64	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 28	Length	40	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 1B	Length	27	Number of bytes in Payload
Payload			
F8	Call Token		Begin Revert Method

Bytes (Hex)	Purpose	Value	Notes
A8 00 00 02 05 00 00 00 01	Short Atom Token	AdminSP_UID	Invoking UID
A8 00 00 00 06 00 00 02 02	Short Atom Token	Revert_UID	Method UID
F0	Start List Token		Begin method parameter list
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00	Pad	0	Included in ComPacket and Packet lengths

3.3.18.2.1 Revert TPer Response

The following pseudo-code is the signature of the response to the `Revert` method of the Admin SP:

```
session[TSN:HSN] <- []
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding from the `Revert` method response (see Table 39 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 2c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 14 00 00 00 00
0030    00 00 00 00 00 00 00 08  f0 f1 f9 f0 00 00 00 f1
0040    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 39: Revert TPer Response

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 2C	Length	44	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 14	Length	20	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 08	Length	8	Number of bytes in Payload
Payload			
F0	Start List Token		Begin results list
F1	End List Token		End results list
F9	End of Data Token		
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

Note: A TPer aborts the session immediately after reporting the `Revert` result.

3.4 KMIP Command Tokenization

3.4.1 Overview

This section provides additional details about the KMIP based commands described in section 3.2, as well as the tokenization of each KMIP message and the packaging of those messages in ComPackets. This section splits the KMIP message into two sections to describe the Compacket Header and the Compacket Payload separately. When

forming a KMIP message, the Compacket Header and payload should be sent over the Key Injection Interface as a single payload as specified in the Key Per I/O SSC [3].

The following is a list of command parameters used in TCG messages with Security Protocol `0x01`. The values for these parameters vary in actual implementations. In this section, the values chosen for these parameters are as follows:

1. All commands use a statically allocated Vendor Unique ComID value of `0x08000000`.
2. The host always uses the Host Session Number (HSN) `0x00000001`.
3. The TPer always uses the TPer Session Number (TSN) `0x00001001`.
4. Communications sent from the host to the TPer have a Packet.SeqNumber of 0.
5. Communications sent from the TPer to the Host have a Packet.SeqNumber of 0.

The following is a command parameter used in KMIP messages with Security Protocol `0x03`. The values for these parameters vary in actual implementations. In this section, the values chosen for these parameters are as follows:

1. KMIP Messages use a statically allocated Vendor Unique ComID value of `0x08010000`.

All transfers between a host and Storage Device are in 512-byte blocks. If the ComPacket does not end at a 512-byte boundary, bytes of `0x00` are appended after the ComPacket to pad the buffer to a 512-byte alignment.

3.4.2 Inject Plaintext KEK

The following subsections describe the procedure a host follows to inject a plaintext KEK into a TPer.

3.4.2.1 Unlock Key Injection Interface

The procedure a host follows to unlock the Key Injection Interface (Security Protocol `0x3`) is described in section 3.3.8.

3.4.2.2 Enable Plaintext KEK Provisioning

After Key Per I/O SP is activated, the default state of the Key column of the `KeyEncryptionKey` Table is empty. A plaintext KEK can be injected into a row in the `KeyEncryptionKey` Table with an empty Key column without enabling plaintext KEK provisioning. If the Key column is not empty for a row in the `KeyEncryptionKey` Table, then either plaintext KEK provisioning needs to be enabled for the particular row (see section 3.3.12) or plaintext KEK provisioning can be enabled for all rows in the `KeyEncryptionKey` Table (see section 3.3.11).

3.4.2.3 Inject Plaintext KEK Request Message

This section describes the procedure a host follows to inject a plaintext KEK into a TPer through the key injection interface. In this example, the KEK injected into the TPer is associated with KeyEncryptionKey1 and the host assigned the KMIP Key UID "c51a6ce0-e11c-4320-80c2-f1f270d2368e" to this KEK (see Table 41). The KMIP Key UID is not discoverable through a Table in the Key Per I/O SP. Therefore, the host will need to track the mapping between the KMIP Key UID and the row in the `KeyEncryptionKey` Table. In this example, the KMIP Key UID "c51a6ce0-e11c-4320-80c2-f1f270d2368e" will be associated with the KeyEncryptionKey1 row.

Security Protocol `0x03` Compacket Header (see Table 40 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
```

```
0010    00 00 01 80 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 40: Inject Plaintext KEK Compacket Header

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 01	ComID	0x0801	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 01 80	Length	384	Number of bytes in Packet

Security Protocol 0x03 Compacket Payload - TTLV byte encoding of Inject KEK Request Message (see Table 41 for a description of the following byte encoding):

```
0000    42 00 78 01 00 00 01 78  42 00 77 01 00 00 00 38
0010    42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020    00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
0030    00 00 00 01 00 00 00 00  42 00 0d 02 00 00 00 04
0040    00 00 00 01 00 00 00 00  42 00 0f 01 00 00 01 30
0050    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
0060    42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0070    42 00 79 01 00 00 01 08  42 00 94 07 00 00 00 24
0080    63 35 31 61 36 63 65 30  2d 65 31 31 63 2d 34 33
0090    32 30 2d 38 30 63 32 2d  66 31 66 32 37 30 64 32
00a0    33 36 38 65 00 00 00 00  42 00 57 05 00 00 00 04
00b0    00 00 00 02 00 00 00 00  42 01 25 01 00 00 00 70
00c0    42 00 2b 01 00 00 00 30  42 00 83 05 00 00 00 04
00d0    00 00 00 0b 00 00 00 00  42 00 28 05 00 00 00 04
00e0    00 00 00 03 00 00 00 00  42 00 2a 02 00 00 00 04
00f0    00 00 01 00 00 00 00 00  42 00 08 01 00 00 00 30
0100    42 00 9d 07 00 00 00 07  54 43 47 2d 53 57 47 00
0110    42 00 0a 07 00 00 00 03  55 49 44 00 00 00 00 00
0120    42 00 0b 08 00 00 00 08  00 00 12 02 00 01 00 01
0130    42 00 8f 01 00 00 00 48  42 00 40 01 00 00 00 40
0140    42 00 42 05 00 00 00 04  00 00 00 01 00 00 00 00
0150    42 00 45 01 00 00 00 28  42 00 43 08 00 00 00 20
0160    00 01 02 03 04 05 06 07  08 09 0a 0b 0c 0d 0e 0f
0170    10 11 12 13 14 15 16 17  18 19 1a 1b 1c 1d 1e 1f
0180    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol 0x03 Compacket Payload - XML encoding of the Inject KEK Request Message:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<KMIP>
    <!-- Inject plaintext KEK into TPer Through Key Injection Interface -->
    <RequestMessage>
        <!-- Request Header -->
        <RequestHeader>
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
            </ProtocolVersion>
            <!-- Note: BatchErrorContinuationOption shall not be present if BatchCount <= 1 -
->
            <!-- Note: BatchOrderOption is optional if BatchCount == 1. BatchOrderOption is
True by default -->
            <!-- Note: BatchCount required -->
            <BatchCount type="Integer" value="1"/>
        </RequestHeader>
        <!-- Batch Item 1: Inject Plaintext KEK for KeyEncryptionKey1 -->
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- Note: UniqueBatchItemID required if BatchCount > 1 -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <RequestPayload>
                <!-- Note: Globally unique KMIP Key UID associated with KeyEncryptionKey1 -->
                <UniqueIdentifier type="TextString" value="c51a6ce0-e11c-4320-80c2-
f1f270d2368e"/>
                <ObjectType type="Enumeration" value="SymmetricKey"/>
                <!-- Note: Specify attributes associated with KEK -->
                <Attributes>
                    <!-- Note: Cryptographic parameters associated with KeyEncryptionKey1 -->
                    <CryptographicParameters>
                        <KeyRoleType type="Enumeration" value="KEK"/>
                        <CryptographicAlgorithm type="Enumeration" value="AES"/>
                        <CryptographicLength type="Integer" value="256"/>
                    </CryptographicParameters>
                    <!-- Note: TCG KEK UID from KeyEncryptionKey table associated with
KeyEncryptionKey1 -->
                    <Attribute>
                        <VendorIdentification type="TextString" value="TCG-SWG"/>
                        <AttributeName type="TextString" value="UID"/>
                        <!-- Note: Specify KeyEncryptionKey1 UID -->
                        <AttributeValue type="ByteString" value="0000120200010001"/>
                    </Attribute>
                </Attributes>
                <!-- Note: Specify KEK key material -->
```

```
                <SymmetricKey>
                    <KeyBlock>
                        <!-- Note: Refer to KMIP User Guide 3.26 for "Raw" Key Format Type --
>
                        <KeyFormatType type="Enumeration" value="Raw"/>
                        <!-- Note: KeyValue is a Structure since key is plaintext -->
                        <!-- Plaintext KEK =
000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F -->
                        <KeyValue>
                            <KeyMaterial type="ByteString"
value="000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F"/>
                        </KeyValue>
                    </KeyBlock>
                </SymmetricKey>
            </RequestPayload>
        </BatchItem>
    </RequestMessage>
</KMIP>
```

Table 41: Inject Plaintext KEK Request Message

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
RequestMessage	Structure		420078	01	00000178	
RequestHeader	Structure		420077	01	00000038	
ProtocolVersion	Structure		420069	01	00000020	
ProtocolVersionMajor	Integer	2	42006A	02	00000004	00000002 00000000
ProtocolVersionMinor	Integer	1	42006B	02	00000004	00000001 00000000
BatchCount	Integer	1	42000D	02	00000004	00000001 00000000
BatchItem	Structure		42000F	01	00000130	
Operation	Enumeration	Import	42005C	05	00000004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	00000001	01 00000000000000
RequestPayload	Structure		420079	01	00000108	
UniqueIdentifier	TextString	c51a6ce0-e11c-4320-80c2-f1f270d2368e	420094	07	00000024	63353161366365302d653131632d343332302d383063322d6631663237306432333633865 00000000
ObjectType	Enumeration	Symmetric Key	420057	05	00000004	00000002 00000000
Attributes	Structure		420125	01	00000070	

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
CryptographicParameters	Structure		42002B	01	00000030	
KeyRoleType	Enumeration	KEK	420083	05	00000004	0000000b 00000000
CryptographicAlgorithm	Enumeration	AES	420028	05	00000004	00000003 00000000
CryptographicLength	Integer	256	42002A	02	00000004	00000100 00000000
Attribute	Structure		420008	01	00000030	
VendorIdentification	TextString	TCG-SWG	42009D	07	00000007	5443472d535747 00
AttributeName	TextString	UID	42000A	07	00000003	554944 0000000000
AttributeValue	ByteString	000012020 0010001	42000B	08	00000008	0000120200010001
SymmetricKey	Structure		42008F	01	00000048	
KeyBlock	Structure		420040	01	00000040	
KeyFormatType	Enumeration	Raw	420042	05	00000004	00000001 00000000
KeyValue	Structure		420045	01	00000028	
KeyMaterial	ByteString	000102030 405060708 090A0B0C 0D0E0F10 111213141 516171819 1A1B1C1D 1E1F	420043	08	00000020	00010203040506070 8090A0B0C0D0E0F10 11121314151617181 91A1B1C1D1E1F

3.4.2.3.1 Inject Plaintext KEK Response Message

Security Protocol 0x03 Compacket Header (see Table 42 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 c8 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 42: Inject Plaintext KEK Response Compacket Header

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 01	ComID	0x0801	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	

Bytes (Hex)	Purpose	Value	Notes
00 00 00 C8	Length	200	Number of bytes in Packet

Security Protocol 0x03 Compacket Payload - TTLV byte encoding of the Inject KEK Response Message:

```
0000    42 00 7b 01 00 00 00 c0  42 00 7a 01 00 00 00 48
0010    42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020    00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
0030    00 00 00 01 00 00 00 00  42 00 92 09 00 00 00 08
0040    00 00 00 00 00 00 00 00  42 00 0d 02 00 00 00 04
0050    00 00 00 01 00 00 00 00  42 00 0f 01 00 00 00 68
0060    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
0070    42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0080    42 00 7f 05 00 00 00 04  00 00 00 00 00 00 00 00
0090    42 00 7c 01 00 00 00 30  42 00 94 07 00 00 00 24
00a0    63 35 31 61 36 63 65 30  2d 65 31 31 63 2d 34 33
00b0    32 30 2d 38 30 63 32 2d  66 31 66 32 37 30 64 32
00c0    33 36 38 65 00 00 00 00  00 00 00 00 00 00 00 00
00d0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol 0x03 Compacket Payload - XML encoding of the Inject KEK Response Message (see Table 43 for a description of the following byte encoding):

```
<KMIP>
    <!-- Inject plaintext KEK Response -->
    <ResponseMessage>
        <ResponseHeader>
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
            </ProtocolVersion>
            <!-- TimeStamp is required -->
            <!-- 0 indicates real-time clock not supported (refer to TCG core 3.3.4.7.4) -->
            <TimeStamp type="DateTime" value="0"/>
            <BatchCount type="Integer" value="1"/>
        </ResponseHeader>
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- UniqueBatchItemID value shall match value in request -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <ResultStatus type="Enumeration" value="Success"/>
            <ResponsePayload>
                <!-- UniqueIdentifier value shall match value in request -->
```

```
                <UniqueIdentifier type="TextString" value="c51a6ce0-e11c-4320-80c2-
f1f270d2368e"/>
            </ResponsePayload>
        </BatchItem>
    </ResponseMessage>
</KMIP>
```

Table 43: Inject Plaintext KEK Response Message

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
ResponseMessage	Structure		42007B	01	000000c0	
ResponseHeader	Structure		42007A	01	00000048	
ProtocolVersion	Structure		420069	01	00000020	
ProtocolVersionMajor	Integer	2	42006A	02	00000004	00000002 00000000
ProtocolVersionMinor	Integer	1	42006B	02	00000004	00000001 00000000
TimeStamp	DateTime	0	420092	09	00000008	00000000 00000000
BatchCount	Integer	1	42000D	02	00000004	00000001 00000000
BatchItem	Structure		42000F	01	00000068	
Operation	Enumeration	Import	42005C	05	00000004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	00000001	01 00000000000000
ResultStatus	Enumeration	Success	42007F	05	00000004	00000000 00000000
ResponsePayload	Structure		42007C	01	00000030	
UniqueIdentifier	TextString	c51a6ce0-e11c-4320-80c2-f1f270d2368e	420094	07	00000024	63353161366365302d653131632d343332302d383063322d663166323730643233363865 00000000

3.4.3 Replace KEK1 with new KEK wrapped with KEK1

The following subsections describe the procedure a host follows to replace an existing KEK in a TPer with a new KEK. The procedure to inject the initial KEK is described in section 3.4.2.

3.4.3.1 Unlock Key Injection Interface

The procedure a host follows to unlock the Key Injection Interface (Security Protocol 0x3) is described in section 3.3.8.

3.4.3.2 Replace KEK Request Message

This section describes the procedure a host follows to inject a wrapped KEK into a TPer through the key injection interface that will replace an existing KEK. In this example, the KEK injected into the TPer is associated with KeyEncryptionKey1 and the wrapping KEK is currently residing within the TPer and the KEK is also associated with KeyEncryptionKey1. The host specifies the KMIP Key UID "c51a6ce0-e11c-4320-80c2-f1f270d2368e" to reference the existing KEK associated with KeyEncryptionKey1 (see section 3.4.2.2) and the host assigns a new KMIP Key UID "aff7a01b-7a5d-4d0f-a3d3-e6a9fd8020b6" (see Table 44 for the ComPacket header and Table 45 for the ComPacket payload).

Security Protocol `0x03` Compacket Header (see Table 44 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
0010    00 00 01 f8 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 44: Replace KEK Compacket Header

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 01	ComID	0x0801	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 01 F8	Length	504	Number of bytes in Packet

Security Protocol `0x03` Compacket Payload - TTLV byte encoding of the Inject KEK Request Message (see Table 45 for a description of the following byte encoding):

```
0000    42 00 78 01 00 00 01 f0  42 00 77 01 00 00 00 38
0010    42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020    00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
0030    00 00 00 01 00 00 00 00  42 00 0d 02 00 00 00 04
0040    00 00 00 01 00 00 00 00  42 00 0f 01 00 00 01 a8
0050    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
0060    42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0070    42 00 79 01 00 00 01 80  42 00 94 07 00 00 00 24
0080    61 66 66 37 61 30 31 62  2d 37 61 35 64 2d 34 64
0090    30 66 2d 61 33 64 33 2d  65 36 61 39 66 64 38 30
00a0    32 30 62 36 00 00 00 00  42 00 57 05 00 00 00 04
00b0    00 00 00 02 00 00 00 00  42 01 25 01 00 00 00 70
00c0    42 00 2b 01 00 00 00 30  42 00 83 05 00 00 00 04
00d0    00 00 00 0b 00 00 00 00  42 00 28 05 00 00 00 04
00e0    00 00 00 03 00 00 00 00  42 00 2a 02 00 00 00 04
00f0    00 00 01 00 00 00 00 00  42 00 08 01 00 00 00 30
0100    42 00 9d 07 00 00 00 07  54 43 47 2d 53 57 47 00
0110    42 00 0a 07 00 00 00 03  55 49 44 00 00 00 00 00
```

```
0120    42 00 0b 08 00 00 00 08  00 00 12 02 00 01 00 01
0130    42 00 8f 01 00 00 00 c0  42 00 40 01 00 00 00 b8
0140    42 00 42 05 00 00 00 04  00 00 00 01 00 00 00 00
0150    42 00 45 08 00 00 00 28  d3 05 b6 04 c8 e8 64 31
0160    e9 41 e0 7c 96 d2 9a 21  c3 26 e1 2f 17 b9 16 96
0170    76 2e 99 60 74 84 02 dd  32 a6 67 88 d5 55 79 66
0180    42 00 46 01 00 00 00 70  42 00 9e 05 00 00 00 04
0190    00 00 00 01 00 00 00 00  42 00 36 01 00 00 00 58
01a0    42 00 94 07 00 00 00 24  63 35 31 61 36 63 65 30
01b0    2d 65 31 31 63 2d 34 33  32 30 2d 38 30 63 32 2d
01c0    66 31 66 32 37 30 64 32  33 36 38 65 00 00 00 00
01d0    42 00 2b 01 00 00 00 20  42 00 28 05 00 00 00 04
01e0    00 00 00 03 00 00 00 00  42 00 11 05 00 00 00 04
01f0    00 00 00 0d 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol `0x03` Compacket Payload - XML encoding of the Inject KEK Request Message:

```
<?xml version="1.0" encoding="UTF-8"?>
<KMIP>
    <!-- Replace existing KEK in KeyEncryptionKey1 with new KEK -->
    <RequestMessage>
        <!-- Request Header -->
        <RequestHeader>
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
            </ProtocolVersion>
            <!-- Note: BatchErrorContinuationOption shall not be present if BatchCount <= 1 -
->
            <!-- Note: BatchOrderOption is optional if BatchCount == 1. BatchOrderOption is
True by default -->
            <!-- Note: BatchCount required -->
            <BatchCount type="Integer" value="1"/>
        </RequestHeader>
        <!-- Batch Item 1: Inject Wrapped KEK -->
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- Note: UniqueBatchItemID required if BatchCount > 1 -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <RequestPayload>
                <!-- Note: Globally unique KMIP Key UID associated with KEK -->
                <UniqueIdentifier type="TextString" value="aff7a01b-7a5d-4d0f-a3d3-
e6a9fd8020b6"/>
                <ObjectType type="Enumeration" value="SymmetricKey"/>
                <Attributes>
                    <!-- Note: Cryptographic parameters associated with KeyEncryptionKey1 -->
                    <CryptographicParameters>
```

```
                    <KeyRoleType type="Enumeration" value="KEK"/>
                    <CryptographicAlgorithm type="Enumeration" value="AES"/>
                    <CryptographicLength type="Integer" value="256"/>
                </CryptographicParameters>
                <!-- Note: TCG KEK UID from KeyEncryptionKey table associated with KEK --
>
                <Attribute>
                    <VendorIdentification type="TextString" value="TCG-SWG"/>
                    <AttributeName type="TextString" value="UID"/>
                    <!-- KeyEncryptionKey1 UID -->
                    <AttributeValue type="ByteString" value="0000120200010001"/>
                </Attribute>
            </Attributes>
            <!-- Note: Specify KEK key material -->
            <SymmetricKey>
                <KeyBlock>
                    <KeyFormatType type="Enumeration" value="Raw"/>
                    <!-- Note: KeyValue is a ByteString since key is wrapped -->
                    <!-- KEK associated with KeyEncryptionKey1 is wrapped with existing
KEK associated with KeyEncryptionKey1 -->
                    <!-- Unwrapped new KEK =
ABC102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F -->
                    <!-- Existing wrapping KEK =
000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F -->
                    <KeyValue type="ByteString"
value="d305b604c8e86431e941e07c96d29a21c326e12f17b91696762e9960748402dd32a66788d5557966"/>
                    <!-- Note: Specify previously injected key use to wrap injected key -
->
                    <KeyWrappingData>
                        <!-- Note: Method used to wrap Key Value -->
                        <WrappingMethod type="Enumeration" value="Encrypt"/>
                        <EncryptionKeyInformation>
                            <!-- Note: KMIP Key UID of KEK used to wrap injected key -->
                            <!-- KMIP Key UID associated with KeyEncryptionKey1 -->
                            <UniqueIdentifier  type="TextString" value="c51a6ce0-e11c-
4320-80c2-f1f270d2368e"/>
                            <CryptographicParameters>
                                <!-- Note: Crypto Algorithm used to wrap key -->
                                <CryptographicAlgorithm type="Enumeration" value="AES"/>
                                <BlockCipherMode type="Enumeration" value="NISTKeyWrap"/>
                            </CryptographicParameters>
                        </EncryptionKeyInformation>
                    </KeyWrappingData>
                </KeyBlock>
            </SymmetricKey>
```

```
            </RequestPayload>
        </BatchItem>
    </RequestMessage>
</KMIP>
```

Table 45: Replace KEK Request Message

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
RequestMessage	Structure		420078	01	0000 01f0	
RequestHeader	Structure		420077	01	0000 0038	
ProtocolVersion	Structure		420069	01	0000 0020	
ProtocolVersionMajor	Integer	2	42006A	02	0000 0004	00000002 00000000
ProtocolVersionMinor	Integer	1	42006B	02	0000 0004	00000001 00000000
BatchCount	Integer	1	42000D	02	0000 0004	00000001 00000000
BatchItem	Structure		42000F	01	0000 01a8	
Operation	Enumeration	Import	42005C	05	0000 0004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	0000 0001	01 00000000000000
RequestPayload	Structure		420079	01	0000 0180	
UniqueIdentifier	TextString	aff7a01b-7a5d-4d0f-a3d3-e6a9fd8020b6	420094	07	0000 0024	61666637613031622d376135642d346430662d613364332d653661396664383032306236 00000000
ObjectType	Enumeration	SymmetricKey	420057	05	0000 0004	00000002 00000000
Attributes	Structure		420125	01	0000 0070	
CryptographicParameters	Structure		42002B	01	0000 0030	
KeyRoleType	Enumeration	KEK	420083	05	0000 0004	0000000b 00000000
CryptographicAlgorithm	Enumeration	AES	420028	05	0000 0004	00000003 00000000

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
CryptographicLength	Integer	256	42002A	02	0000 0004	00000100 00000000
Attribute	Structure		420008	01	0000 0030	
VendorIdentification	TextString	TCG-SWG	42009D	07	0000 0007	5443472d535747 00
AttributeName	TextString	UID	42000A	07	0000 0003	554944 0000000000
AttributeValue	ByteString	0000120200010001	42000B	08	0000 0008	0000120200010001
SymmetricKey	Structure		42008F	01	0000 00c0	
KeyBlock	Structure		420040	01	0000 00b8	
KeyFormatType	Enumeration	Raw	420042	05	0000 0004	00000001 00000000
KeyValue	ByteString	d305b604c8e86431e941e07c96d29a21c326e12f17b91696762e9960748402dd32a66788d5557966	420045	08	0000 0028	d305b604c8e86431e941e07c96d29a21c326e12f17b91696762e9960748402dd32a66788d5557966
KeyWrappingData	Structure		420046	01	0000 0070	
WrappingMethod	Enumeration	Encrypt	42009E	05	0000 0004	00000001 00000000
EncryptionKeyInformation	Structure		420036	01	0000 0058	
UniqueIdentifier	TextString	c51a6ce0-e11c-4320-80c2-f1f270d2368e	420094	07	0000 0024	63353161366365302d653131632d343332302d383063322d663166323730643233363865 00000000
CryptographicParameters	Structure		42002B	01	0000 0020	
CryptographicAlgorithm	Enumeration	AES	420028	05	0000 0004	00000003 00000000
BlockCipherMode	Enumeration	NISTKeyWrap	420011	05	0000 0004	0000000d 00000000

3.4.3.2.1 Replace KEK Response Message

Security Protocol 0x03 Compacket Header (see Table 46 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 c8 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 46: Replace KEK Response Compacket Header

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 01	ComID	0x0801	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 C8	Length	200	Number of bytes in Packet

Security Protocol 0x03 Compacket Payload - TTLV byte encoding of the Inject KEK Request Message (see Table 47 for a description of the following byte encoding):

```
0000    42 00 7b 01 00 00 00 c0  42 00 7a 01 00 00 00 48
0010    42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020    00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
0030    00 00 00 01 00 00 00 00  42 00 92 09 00 00 00 08
0040    00 00 00 00 00 00 00 00  42 00 0d 02 00 00 00 04
0050    00 00 00 01 00 00 00 00  42 00 0f 01 00 00 00 68
0060    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
0070    42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0080    42 00 7f 05 00 00 00 04  00 00 00 00 00 00 00 00
0090    42 00 7c 01 00 00 00 30  42 00 94 07 00 00 00 24
00a0    61 66 66 37 61 30 31 62  2d 37 61 35 64 2d 34 64
00b0    30 66 2d 61 33 64 33 2d  65 36 61 39 66 64 38 30
00c0    32 30 62 36 00 00 00 00  00 00 00 00 00 00 00 00
00d0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol 0x03 Compacket Payload - XML encoding of the Inject KEK Request Message:

```
<KMIP>
    <!-- Replace existing KEK Response -->
    <ResponseMessage>
        <ResponseHeader>
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
            </ProtocolVersion>
```

```
            <!-- TimeStamp is required -->
            <!-- 0 indicates real-time clock not supported (refer to TCG core 3.3.4.7.4) -->
            <TimeStamp type="DateTime" value="0"/>
            <BatchCount type="Integer" value="1"/>
        </ResponseHeader>
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- UniqueBatchItemID value shall match value in request -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <ResultStatus type="Enumeration" value="Success"/>
            <ResponsePayload>
                <!-- UniqueIdentifier value shall match value in request -->
                <UniqueIdentifier type="TextString" value="aff7a01b-7a5d-4d0f-a3d3-
e6a9fd8020b6"/>
            </ResponsePayload>
        </BatchItem>
    </ResponseMessage>
</KMIP>
```

Table 47: Replace KEK Response Message

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
ResponseMessage	Structure		42007B	01	000000c0	
ResponseHeader	Structure		42007A	01	00000048	
ProtocolVersion	Structure		420069	01	00000020	
ProtocolVersionMajor	Integer	2	42006A	02	00000004	00000002 00000000
ProtocolVersionMinor	Integer	1	42006B	02	00000004	00000001 00000000
TimeStamp	DateTime	0	420092	09	00000008	00000000 00000000
BatchCount	Integer	1	42000D	02	00000004	00000001 00000000
BatchItem	Structure		42000F	01	00000068	
Operation	Enumeration	Import	42005C	05	00000004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	00000001	01 00000000000000
ResultStatus	Enumeration	Success	42007F	05	00000004	00000000 00000000
ResponsePayload	Structure		42007C	01	00000030	
UniqueIdentifier	TextString	aff7a01b-7a5d-4d0f-a3d3-e6a9fd8020b6	420094	07	00000024	61666637613031622d376135642d346430662d613364332d6536613966643830323062360 00000000

3.4.4 Inject KEK2 Wrapped with KEK1

The following subsections describe the procedure a host follows to inject a KEK wrapped with an existing KEK into a TPer. The procedure to inject the initial KEK is described in section 3.4.2.

3.4.4.1 Unlock Key Injection Interface

The procedure a host follows to unlock the Key Injection Interface (Security Protocol `0x3`) is described in section 3.3.8.

3.4.4.2 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.4.4.3 Enable Wrapping KeyEncryptionKey2 with KeyEncryptionKey1

```
session[TSN:HSN] -> KeyEncryptionKey2_UID.Set[Values = [AllowedKeyEncryptionKeys
=[KeyEncryptionKey1_UID, KeyEncryptionKey2_UID]]]
```

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 5c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 44 00 00 00 00
0030    00 00 00 00 00 00 00 37  f8 a8 00 00 12 02 00 01
0040    00 02 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    06 f0 a8 00 00 12 02 00  01 00 01 a8 00 00 12 02
0060    00 01 00 02 f1 f3 f1 f3  f1 f9 f0 00 00 00 f1 00
0070    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 48: Enable Wrapping KEK2 with KEK1

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
`00 00 00 00`	Reserved	0	
`08 00`	ComID	`0x0800`	This field is vendor unique
`00 00`	ComIDExtension	0	This field is vendor unique
`00 00 00 00`	OutstandingData	0	
`00 00 00 00`	MinTransfer	0	
`00 00 00 5C`	Length	92	Number of bytes in Packet
Packet			
`00 00 10 01`	TPerSN	`0x1001`	TPer Session Number
`00 00 00 01`	HostSN	1	Host Session Number

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 44	Length	68	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 37	Length	55	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 02 00 01 00 02	Short Atom Token	Key_Encryption_Key2_UID	KeyEncryptionKey UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
06	Tiny Atom Token	6	AllowedKeyEncryptionKeys Column
F0	Start List Token		Begin kek_obj_uidref_list list
A8 00 00 12 02 00 01 00 01	Short Atom Token	Key_Encryption_Key1_UID	KeyEncryptionKey1 UID
A8 00 00 12 02 00 01 00 02	Short Atom Token	Key_Encryption_Key2_UID	KeyEncryptionKey2 UID
F1	End List Token		End kek_obj_uidref_list list
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair

Bytes (Hex)	Purpose	Value	Notes
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00	Pad	0	Included in ComPacket and Packet lengths

The response to the above `Set` method request is a `Set` method response (see section 3.3.3.3).

3.4.4.4 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.4.4.5 Inject Wrapped KEK Request Message

This section describes the procedure a host follows to inject a wrapped KEK into a TPer through the key injection interface. In this example, the KEK injected into the TPer is associated with KeyEncryptionKey2 and the wrapping KEK, which is associated with KeyEncryptionKey1, is currently residing within the TPer. The host specifies the KMIP Key UID "c51a6ce0-e11c-4320-80c2-f1f270d2368e" to reference the existing KEK associated with KeyEncryptionKey1 which was previously injected into the TPer (see section 3.4.2.2). The host assigns a new KMIP Key UID "a8a5842e-abcd-1234-5678-3bfb43cfe569" for the KEK associated with KeyEncryptionKey2 (see Table 49 for the ComPacket header and Table 50 for the ComPacket payload).

Security Protocol `0x03` Compacket Header (see Table 49 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
0010    00 00 01 f8 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 49: Inject Wrapped KEK Compacket Header

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	

Bytes (Hex)	Purpose	Value	Notes
08 01	ComID	0x0801	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 01 F8	Length	504	Number of bytes in Packet

Security Protocol 0x03 Compacket Payload - TTLV byte encoding of the Inject wrapped KEK Request Message (see Table 50 for a description of the following byte encoding):

```
0000    42 00 78 01 00 00 01 f0  42 00 77 01 00 00 00 38
0010    42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020    00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
0030    00 00 00 01 00 00 00 00  42 00 0d 02 00 00 00 04
0040    00 00 00 01 00 00 00 00  42 00 0f 01 00 00 01 a8
0050    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
0060    42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0070    42 00 79 01 00 00 01 80  42 00 94 07 00 00 00 24
0080    61 38 61 35 38 34 32 65  2d 61 62 63 64 2d 31 32
0090    33 34 2d 35 36 37 38 2d  33 62 66 62 34 33 63 66
00a0    65 35 36 39 00 00 00 00  42 00 57 05 00 00 00 04
00b0    00 00 00 02 00 00 00 00  42 01 25 01 00 00 00 70
00c0    42 00 2b 01 00 00 00 30  42 00 83 05 00 00 00 04
00d0    00 00 00 0b 00 00 00 00  42 00 28 05 00 00 00 04
00e0    00 00 00 03 00 00 00 00  42 00 2a 02 00 00 00 04
00f0    00 00 01 00 00 00 00 00  42 00 08 01 00 00 00 30
0100    42 00 9d 07 00 00 00 07  54 43 47 2d 53 57 47 00
0110    42 00 0a 07 00 00 00 03  55 49 44 00 00 00 00 00
0120    42 00 0b 08 00 00 00 08  00 00 12 02 00 01 00 02
0130    42 00 8f 01 00 00 00 c0  42 00 40 01 00 00 00 b8
0140    42 00 42 05 00 00 00 04  00 00 00 01 00 00 00 00
0150    42 00 45 08 00 00 00 28  9f cd 34 14 56 42 b5 fb
0160    da 5c 2f b1 17 de 7b 19  71 f9 be c2 f7 aa c7 b0
0170    49 7f 0a 26 38 0a 4d 77  1d 12 0d d7 3b b8 da 84
0180    42 00 46 01 00 00 00 70  42 00 9e 05 00 00 00 04
0190    00 00 00 01 00 00 00 00  42 00 36 01 00 00 00 58
01a0    42 00 94 07 00 00 00 24  63 35 31 61 36 63 65 30
01b0    2d 65 31 31 63 2d 34 33  32 30 2d 38 30 63 32 2d
01c0    66 31 66 32 37 30 64 32  33 36 38 65 00 00 00 00
01d0    42 00 2b 01 00 00 00 20  42 00 28 05 00 00 00 04
01e0    00 00 00 03 00 00 00 00  42 00 11 05 00 00 00 04
01f0    00 00 00 0d 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol 0x03 Compacket Payload - XML encoding of the Inject wrapped KEK Request Message:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<KMIP>
    <!-- Inject wrapped KEK into TPer Through KMIP -->
    <RequestMessage>
```

```
        <!-- Request Header -->
        <RequestHeader>
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
            </ProtocolVersion>
            <!-- Note: BatchErrorContinuationOption shall not be present if BatchCount <= 1 -
->
            <!-- Note: BatchOrderOption is optional if BatchCount == 1. BatchOrderOption is
True by default -->
            <!-- Note: BatchCount required -->
            <BatchCount type="Integer" value="1"/>
        </RequestHeader>
        <!-- Batch Item 1: Inject Wrapped KEK -->
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- Note: UniqueBatchItemID required if BatchCount > 1 -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <RequestPayload>
                <!-- Note: Globally unique KMIP Key UID associated with KEK -->
                <UniqueIdentifier type="TextString" value="a8a5842e-abcd-1234-5678-
3bfb43cfe569"/>
                <ObjectType type="Enumeration" value="SymmetricKey"/>
                <Attributes>
                    <!-- Note: Cryptographic parameters associated with KeyEncryptionKey2 -->
                    <CryptographicParameters>
                        <KeyRoleType type="Enumeration" value="KEK"/>
                        <CryptographicAlgorithm type="Enumeration" value="AES"/>
                        <CryptographicLength type="Integer" value="256"/>
                    </CryptographicParameters>
                    <!-- Note: TCG KEK UID from KeyEncryptionKey table associated with KEK --
>
                    <Attribute>
                        <VendorIdentification type="TextString" value="TCG-SWG"/>
                        <AttributeName type="TextString" value="UID"/>
                        <!-- KeyEncryptionKey2 UID -->
                        <AttributeValue type="ByteString" value="0000120200010002"/>
                    </Attribute>
                </Attributes>
                <!-- Note: Specify KEK key material -->
                <SymmetricKey>
                    <KeyBlock>
                        <KeyFormatType type="Enumeration" value="Raw"/>
                        <!-- Note: KeyValue is a ByteString since key is wrapped -->
```

```
                        <!-- KEK associated with KeyEncryptionKey2 is wrapped with existing
KEK associated with KeyEncryptionKey1 -->
                        <!-- Unwrapped new KEK =
DEF102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F -->
                        <!-- Existing wrapping KEK =
000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F -->
                        <KeyValue type="ByteString"
value="9fcd34145642b5fbda5c2fb117de7b1971f9bec2f7aac7b0497f0a26380a4d771d120dd73bb8da84"/>
                        <!-- Note: Specify previously injected key use to wrap injected key -
->
                        <KeyWrappingData>
                            <!-- Note: Method used to wrap Key Value -->
                            <WrappingMethod type="Enumeration" value="Encrypt"/>
                            <EncryptionKeyInformation>
                                <!-- Note: KMIP Key UID of KEK used to wrap injected key -->
                                <!-- KMIP Key UID associated with KeyEncryptionKey1 -->
                                <UniqueIdentifier  type="TextString" value="c51a6ce0-e11c-
4320-80c2-f1f270d2368e"/>
                                <CryptographicParameters>
                                    <!-- Note: Crypto Algorithm used to wrap key -->
                                    <CryptographicAlgorithm type="Enumeration" value="AES"/>
                                    <BlockCipherMode type="Enumeration" value="NISTKeyWrap"/>
                                </CryptographicParameters>
                            </EncryptionKeyInformation>
                        </KeyWrappingData>
                    </KeyBlock>
                </SymmetricKey>
            </RequestPayload>
        </BatchItem>
    </RequestMessage>
</KMIP>
```

Table 50: Inject Wrapped KEK Request Message

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
RequestMessage	Structure		420078	01	0000 01f0	
RequestHeader	Structure		420077	01	0000 0038	
ProtocolVersion	Structure		420069	01	0000 0020	
ProtocolVersionMajor	Integer	2	42006A	02	0000 0004	00000002 00000000

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
ProtocolVersionMinor	Integer	1	42006B	02	0000 0004	00000001 00000000
BatchCount	Integer	1	42000D	02	0000 0004	00000001 00000000
BatchItem	Structure		42000F	01	0000 01a8	
Operation	Enumeration	Import	42005C	05	0000 0004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	0000 0001	01 00000000000000
RequestPayload	Structure		420079	01	0000 0180	
UniqueIdentifier	TextString	a8a5842e-abcd-1234-5678-3bfb43cfe569	420094	07	0000 0024	613861353834326 52d616263642d313233342d353637382d336266623433636665353639 00000000
ObjectType	Enumeration	SymmetricKey	420057	05	0000 0004	00000002 00000000
Attributes	Structure		420125	01	0000 0070	
CryptographicParameters	Structure		42002B	01	0000 0030	
KeyRoleType	Enumeration	KEK	420083	05	0000 0004	0000000b 00000000
CryptographicAlgorithm	Enumeration	AES	420028	05	0000 0004	00000003 00000000
CryptographicLength	Integer	256	42002A	02	0000 0004	00000100 00000000
Attribute	Structure		420008	01	0000 0030	
VendorIdentification	TextString	TCG-SWG	42009D	07	0000 0007	5443472d535747 00
AttributeName	TextString	UID	42000A	07	0000 0003	554944 0000000000
AttributeValue	ByteString	0000120200010002	42000B	08	0000 0008	0000120200010002
SymmetricKey	Structure		42008F	01	0000 00c0	
KeyBlock	Structure		420040	01	0000 00b8	

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
KeyFormatType	Enumeration	Raw	420042	05	0000 0004	00000001 00000000
KeyValue	ByteString	9fcd34145642 b5fbda5c2fb11 7de7b1971f9b ec2f7aac7b04 97f0a26380a4 d771d120dd73 bb8da84	420045	08	0000 0028	9fcd34145642b5fbda5 c2fb117de7b1971f9be c2f7aac7b0497f0a263 80a4d771d120dd73bb8 da84
KeyWrappingData	Structure		420046	01	0000 0070	
WrappingMethod	Enumeration	Encrypt	42009E	05	0000 0004	00000001 00000000
EncryptionKeyInformation	Structure		420036	01	0000 0058	
UniqueIdentifier	TextString	c51a6ce0- e11c-4320- 80c2- f1f270d2368e	420094	07	0000 0024	63353161366365302d6 53131632d343332302d 383063322d663166323 730643233363865 00000000
CryptographicParameters	Structure		42002B	01	0000 0020	
CryptographicAlgorithm	Enumeration	AES	420028	05	0000 0004	00000003 00000000
BlockCipherMode	Enumeration	NISTKeyWrap	420011	05	0000 0004	0000000d 00000000

3.4.4.5.1 Inject Wrapped KEK Response Message

Security Protocol 0x03 Compacket Header (see Table 51 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 c8 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 51: Inject Wrapped KEK Response Compacket Header

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 01	ComID	0x0801	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 C8	Length	200	Number of bytes in Packet

Security Protocol 0x03 Compacket Payload - TTLV byte encoding of the Inject wrapped KEK Request Message (see Table 52 for a description of the following byte encoding):

```
0000    42 00 7b 01 00 00 00 c0  42 00 7a 01 00 00 00 48
0010    42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020    00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
0030    00 00 00 01 00 00 00 00  42 00 92 09 00 00 00 08
0040    00 00 00 00 00 00 00 00  42 00 0d 02 00 00 00 04
0050    00 00 00 01 00 00 00 00  42 00 0f 01 00 00 00 68
0060    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
0070    42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0080    42 00 7f 05 00 00 00 04  00 00 00 00 00 00 00 00
0090    42 00 7c 01 00 00 00 30  42 00 94 07 00 00 00 24
00a0    61 38 61 35 38 34 32 65  2d 61 62 63 64 2d 31 32
00b0    33 34 2d 35 36 37 38 2d  33 62 66 62 34 33 63 66
00c0    65 35 36 39 00 00 00 00  00 00 00 00 00 00 00 00
00d0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol 0x03 Compacket Payload - XML encoding of the Inject wrapped KEK Request Message:

```
<KMIP>
    <!-- Inject wrapped KEK Response -->
    <ResponseMessage>
        <ResponseHeader>
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
            </ProtocolVersion>
            <!-- TimeStamp is required -->
            <!-- 0 indicates real-time clock not supported (refer to TCG core 3.3.4.7.4) -->
            <TimeStamp type="DateTime" value="0"/>
            <BatchCount type="Integer" value="1"/>
        </ResponseHeader>
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- UniqueBatchItemID value shall match value in request -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <ResultStatus type="Enumeration" value="Success"/>
```

```
            <ResponsePayload>
                <!-- UniqueIdentifier value shall match value in request -->
                <UniqueIdentifier type="TextString" value="a8a5842e-abcd-1234-5678-
3bfb43cfe569"/>
            </ResponsePayload>
        </BatchItem>
    </ResponseMessage>
</KMIP>
```

Table 52: Inject Wrapped KEK Response Message

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
ResponseMessage	Structure		42007B	01	000000c0	
ResponseHeader	Structure		42007A	01	00000048	
ProtocolVersion	Structure		420069	01	00000020	
ProtocolVersionMajor	Integer	2	42006A	02	00000004	00000002 00000000
ProtocolVersionMinor	Integer	1	42006B	02	00000004	00000001 00000000
TimeStamp	DateTime	0	420092	09	00000008	00000000 00000000
BatchCount	Integer	1	42000D	02	00000004	00000001 00000000
BatchItem	Structure		42000F	01	00000068	
Operation	Enumeration	Import	42005C	05	00000004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	00000001	01 00000000000000
ResultStatus	Enumeration	Success	42007F	05	00000004	00000000 00000000
ResponsePayload	Structure		42007C	01	00000030	
UniqueIdentifier	TextString	a8a5842e-abcd-1234-5678-3bfb43cfe569	420094	07	00000024	61386135383432652d616263642d313233342d353637382d3362662343363666535 3639 00000000

3.4.5 Inject KEK Wrapped with Public Key

The following subsections describe the procedure a host follows to inject a KEK wrapped with a public key into a TPer. The public key is extracted from a certificate residing in the TPer and the procedure to retrieve the certificate is described in the following subsections.

3.4.5.1 Unlock Key Injection Interface

The procedure a host follows to unlock the Key Injection Interface (Security Protocol 0x3) is described in section 3.3.8.

3.4.5.2 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.4.5.3 Enable Injecting KEK wrapped with Public Key

The procedure a host follows to enable injection of a KEK wrapped with a public key is described in section 3.3.13 and section 3.3.14.

3.4.5.4 GET Certificate Name, UID, and Size from Certificate Table

This section describes the procedure a host follows to retrieve the Certificate Name, `KeyPerIOPublicKeyCertificateData` byte table UID, and the size of the certificate data in the `KeyPerIOPublicKeyCertificateData` byte table (see Table 53). The response payload (see Table 54) contains the Certificate row Name, Common Name, `KeyPerIOPublicKeyCertificateData` UID [3], and the size of the certificate data in the `KeyPerIOPublicKeyCertificateData` byte table. The size of the certificate data in the `KeyPerIOPublicKeyCertificateData` byte table is vendor unique.

The following pseudo-code is the signature of the `Get` method used to retrieve the Name, CommonName, CertData, and CertSize columns of the Certificate1 row in the `Certificates` Table:

```
session[TSN:HSN] -> Certificate1_UID.Get[Cellblock : [startColumn = Name, endColumn = CertSize]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding of an invocation of the `Get` method on Certicate1 row (see Table 53 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 4c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 34 00 00 00 00
0030    00 00 00 00 00 00 00 25  f8 a8 00 00 00 0a 00 00
0040    00 01 a8 00 00 00 06 00  00 00 16 f0 f0 f2 03 03
0050    f3 f2 04 04 f3 f1 f1 f9  f0 00 00 00 f1 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 53: Get Certificate Name, UID, and Size of Certificate Data

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
`00 00 00 00`	Reserved	0	
`08 00`	ComID	`0x0800`	This field is vendor unique
`00 00`	ComIDExtension	0	This field is vendor unique
`00 00 00 00`	OutstandingData	0	

Bytes (Hex)	Purpose	Value	Notes
00 00 00 00	MinTransfer	0	
00 00 00 4C	Length	76	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 34	Length	52	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 25	Length	37	Number of bytes in Payload
Payload			
F8	Call Token		Begin Get Method
A8 00 00 00 0A 00 00 00 01	Short Atom Token	Certificate1_UID	Certificate1 UID
A8 00 00 00 06 00 00 00 16	Short Atom Token	GetMethod_UID	Get Method UID
F0	Start List Token		Begin method parameter list
F0	Start List Token		Begin cell block
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	Start Column
03	Tiny Atom Token	3	CertData Column
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
04	Tiny Atom Token	4	End Column
04	Tiny Atom Token	4	CertSize Column
F3	End Name Token		End Name-Value pair

Bytes (Hex)	Purpose	Value	Notes
F1	End List Token		End cell block
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00 00 00	Pad	0	Included in ComPacket and Packet lengths

3.4.5.4.1 Response

The following pseudo-code is the signature of the `Get` method response that contains the Name, CommonName, CertData, and CertSize columns of the Certificate1 row in the `Certificates` Table:

```
session[TSN:HSN] <- [ [Name = "Certificate1", CommonName = "", CertData =
KeyPerIOPublicKeyCertificateData_UID, CertSize = 750] ]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding of the response to the `Get` method on Certicate1 row (see Table 54 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 44 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 2c 00 00 00 00
0030    00 00 00 00 00 00 00 1e  f0 f0 f2 03 a8 00 00 12
0040    04 00 00 00 00 f3 f2 04  84 00 00 02 ee f3 f1 f1
0050    f9 f0 00 00 00 f1 00 00  00 00 00 00 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 54: Get Certificate UID and Size of Certificate Response

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique

Bytes (Hex)	Purpose	Value	Notes
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 44	Length	68	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 2C	Length	44	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 1E	Length	30	Number of bytes in Payload
Payload			
F0	Start List Token		Begin results list
F0	Start List Token		Begin first row of results
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	CertData Column
A8 00 00 12 04 00 00 00 00	Short Atom Token	0x0000120400000000	KeyPerIOPublicKeyCertificateData byte table UID
F3	End Name Token		End Name-Value pair

Bytes (Hex)	Purpose	Value	Notes
F2	Start Name Token		Begin Name-Value pair
04	Tiny Atom Token	4	CertSize Column
84 00 00 02 EE	Short Atom Token	750	KeyPerIOPublicKeyCertificateData byte table size - This field is vendor unique
F3	End Name Token		End Name-Value pair
F1	End List Token		End first row of results
F1	End List Token		End results list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list
00 00	Pad	0	Included in ComPacket and Packet lengths

3.4.5.5 GET Certificate data from KeyPerIOPublicKeyCertificateData byte table

This section describes the procedure a host follows to retrieve the certificate data contained in the `KeyPerIOPublicKeyCertificateData` byte table (see Table 55). Note: the UID of the `KeyPerIOPublicKeyCertificateData` byte table and the size of the data to retrieve from the byte table can be obtained using the method described in section 3.4.5.4. The response payload (see Table 56) contains the entire certificate from the `KeyPerIOPublicKeyCertificateData` byte table. The data contained in the `KeyPerIOPublicKeyCertificateData` byte table is vendor unique.

The following pseudo-code is the signature of the `Get` method used to retrieve the certificate data contained in the `KeyPerIOPublicKeyCertificateData` byte table:

```
session[TSN:HSN] -> KeyPerIOPublicKeyCertificateData_UID.Get[Cellblock : [startRow =
0, endRow = 749]]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding of the invocation of the `Get` method on Public Key Certificate byte table (see Table 55 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 50 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 38 00 00 00 00
0030    00 00 00 00 00 00 00 29  f8 a8 00 00 12 04 00 00
0040    00 00 a8 00 00 00 06 00  00 00 16 f0 f0 f2 01 00
0050    f3 f2 02 84 00 00 02 ed  f3 f1 f1 f9 f0 00 00 00
```

```
0060     f1 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
0070     00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
...
01f0     00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
```

Table 55: Get Certificate Data

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 50	Length	80	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 38	Length	56	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket

Bytes (Hex)	Purpose	Value	Notes
00 00 00 29	Length	41	Number of bytes in Payload
Payload			
F8	Call Token		Begin Get Method
A8 00 00 12 04 00 00 00 00	Short Atom Token	KeyPerIOPublicKeyCertificateData_UID	KeyPerIOPublicKeyCertificateData UID
A8 00 00 00 06 00 00 00 16	Short Atom Token	GetMethod_UID	Get Method UID
F0	Start List Token		Begin method parameter list
F0	Start List Token		Begin cell block
F2	Start Name Token		Begin Name-Value pair
01	Tiny Atom Token	1	Start Row
00	Tiny Atom Token	0	First Row of Certificate data
F3	End Name Token		End Name-Value pair
F2	Start Name Token		Begin Name-Value pair
02	Tiny Atom Token	2	End Row
84 00 00 02 ED	Short Atom Token	749	Last Row of Certificate data
F3	End Name Token		End Name-Value pair
F1	End List Token		End cell block
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved

Bytes (Hex)	Purpose	Value	Notes
F1	End List Token		End Method Status list
00 00 00	Pad	0	Included in ComPacket and Packet lengths

3.4.5.5.1 Response

The following pseudo-code is the signature of the `Get` method response that contains the certificate data from the `KeyPerIOPublicKeyCertificateData` byte table:

```
session[TSN:HSN] <- [“<certificate data stored in KeyPerIOPublicKeyCertificateData
>”]
```

Security Protocol `0x01` ComPacket Payload – TCG Method byte encoding of the response to the `Get` method on Public Key Certificate byte table (see Table 56 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 03 1c 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 03 04 00 00 00 00
0030    00 00 00 00 00 00 02 f7  f0 d2 ed 30 82 02 ea 30
0040    82 01 d2 a0 03 02 01 02  02 14 4d 77 be e6 9a c9
0050    0d 67 7c 30 a2 16 97 69  68 00 38 c2 b3 7a 30 0d
0060    06 09 2a 86 48 86 f7 0d  01 01 0b 05 00 30 1b 31
0070    19 30 17 06 03 55 04 03  0c 10 44 65 76 69 63 65
0080    20 56 65 6e 64 6f 72 20  43 41 30 20 17 0d 32 31
0090    31 31 33 30 31 35 32 34  35 33 5a 18 0f 32 30 35
00a0    32 31 31 32 32 31 35 32  34 35 33 5a 30 1a 31 18
00b0    30 16 06 03 55 04 03 0c  0f 4b 65 79 20 50 65 72
00c0    20 49 2f 4f 20 50 4b 49  30 82 01 22 30 0d 06 09
00d0    2a 86 48 86 f7 0d 01 01  01 05 00 03 82 01 0f 00
00e0    30 82 01 0a 02 82 01 01  00 e9 53 56 10 6f d6 97
00f0    78 c2 c7 aa 62 8e 80 fb  d0 b8 14 04 38 67 1f a0
0100    7f 2e 81 94 d8 3b 0c e8  41 fa 0b 2b a7 90 b1 59
0110    92 b4 98 80 ea 45 65 2c  a4 ed b7 04 fa ef ee 5a
0120    da 1f 1e 9d b6 d5 8f 3d  8c 90 35 7f 24 b6 1b 00
0130    86 7c 3b 4c 37 7b a9 37  d6 79 aa 54 d1 ad de d6
0140    60 46 33 12 c9 91 b6 d3  02 88 cb d4 3a 7b 08 94
0150    e1 a0 d9 66 1e 73 6d 11  4e 0e 87 48 37 a2 ea f2
0160    2c de 1c cd f0 f4 8b c5  83 ba 02 de 06 b7 c5 39
0170    e7 2b 8e b6 d5 ec 37 82  ae 29 4e 4d 12 e5 0f de
0180    5d bc 0d 4a b7 bd bd 27  79 2d fa 3a 4b bd ab e4
0190    52 39 f3 84 bb bd 7c 3d  9c 6d 0a d4 c4 98 39 d3
01a0    5a a7 ed 7a 3c 26 db aa  cb 31 fd dd ff 72 d2 6f
01b0    33 8a b4 7a d8 bd 92 03  27 6f 52 2a 62 3b 0f 28
01c0    e7 ea 1a c2 4f 0f c4 3c  76 85 d7 60 e0 aa 7c cc
01d0    31 d0 ef a4 e6 19 f8 fa  03 8b 5c 01 f4 43 2c 8d
01e0    58 b7 41 93 ff 4b 5d e4  83 02 03 01 00 01 a3 25
01f0    30 23 30 13 06 03 55 1d  11 04 0c 30 0a 82 08 44
0200    4e 53 20 4e 61 6d 65 30  0c 06 03 55 1d 13 01 01
0210    ff 04 02 30 00 30 0d 06  09 2a 86 48 86 f7 0d 01
0220    01 0b 05 00 03 82 01 01  00 91 72 ce a1 f5 17 3b
0230    07 f3 83 9d d5 87 54 0b  1f f7 5f 74 7a 8e ad 20
```

```
0240    e7 66 e3 0d 67 5e f9 0a  3c a0 05 d5 62 11 1f 40
0250    cc 02 30 9d a1 b3 a1 b6  f6 96 7d 47 18 d2 19 91
0260    f0 cd ec c6 2f 39 fb 25  3d 5c 27 5e c4 e8 d4 45
0270    58 e2 50 75 ca 02 a3 27  3a 62 2a 12 e3 99 8a de
0280    82 b7 08 05 8d 8c 29 5b  3d e5 0f a9 d5 66 6d 78
0290    34 91 e4 cc 06 48 af f8  75 f9 bf 85 55 1d b6 fb
02a0    df 96 eb 3f d5 e5 85 de  7d e0 05 13 65 24 ea 8b
02b0    b0 ac 29 87 5e 6c 73 eb  78 47 b8 6a 29 69 f3 ca
02c0    4b f1 d4 67 93 cf 0b 8a  7a 97 b7 92 1c 53 42 98
02d0    37 f6 a4 53 64 39 91 fc  45 dd b4 7b 62 17 30 07
02e0    5e c1 ca 6b ec b8 c9 8a  91 80 21 25 e5 49 43 f5
02f0    ef 40 d6 37 5a 20 85 c5  5b c8 27 4d 8a 4e e5 cc
0300    a3 f1 a9 4b 76 9f 6b a1  aa 6e bd ff 73 f0 46 67
0310    81 e1 23 47 1b aa 3b 8f  02 78 16 00 42 28 c8 58
0320    e4 3e 21 2b 3a 8c 07 a3  e0 f1 f9 f0 00 00 00 f1
0330    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
03f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

The following describes the details of the public certificate contained in the response payload. The data was generated using the tool OpenSSL. The use of the tool OpenSSL is not required. Note: the attributes and data contained in the public key certificate as show below are vendor unique.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            4d:77:be:e6:9a:c9:0d:67:7c:30:a2:16:97:69:68:00:38:c2:b3:7a
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = Device Vendor CA
        Validity
            Not Before: Nov 30 15:24:53 2021 GMT
            Not After : Nov 22 15:24:53 2052 GMT
        Subject: CN = Key Per I/O PKI
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:e9:53:56:10:6f:d6:97:78:c2:c7:aa:62:8e:80:
                    fb:d0:b8:14:04:38:67:1f:a0:7f:2e:81:94:d8:3b:
                    0c:e8:41:fa:0b:2b:a7:90:b1:59:92:b4:98:80:ea:
                    45:65:2c:a4:ed:b7:04:fa:ef:ee:5a:da:1f:1e:9d:
                    b6:d5:8f:3d:8c:90:35:7f:24:b6:1b:00:86:7c:3b:
                    4c:37:7b:a9:37:d6:79:aa:54:d1:ad:de:d6:60:46:
                    33:12:c9:91:b6:d3:02:88:cb:d4:3a:7b:08:94:e1:
                    a0:d9:66:1e:73:6d:11:4e:0e:87:48:37:a2:ea:f2:
                    2c:de:1c:cd:f0:f4:8b:c5:83:ba:02:de:06:b7:c5:
                    39:e7:2b:8e:b6:d5:ec:37:82:ae:29:4e:4d:12:e5:
                    0f:de:5d:bc:0d:4a:b7:bd:bd:27:79:2d:fa:3a:4b:
```

```
                bd:ab:e4:52:39:f3:84:bb:bd:7c:3d:9c:6d:0a:d4:
                c4:98:39:d3:5a:a7:ed:7a:3c:26:db:aa:cb:31:fd:
                dd:ff:72:d2:6f:33:8a:b4:7a:d8:bd:92:03:27:6f:
                52:2a:62:3b:0f:28:e7:ea:1a:c2:4f:0f:c4:3c:76:
                85:d7:60:e0:aa:7c:cc:31:d0:ef:a4:e6:19:f8:fa:
                03:8b:5c:01:f4:43:2c:8d:58:b7:41:93:ff:4b:5d:
                e4:83
            Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:DNS Name
            X509v3 Basic Constraints: critical
                CA:FALSE
    Signature Algorithm: sha256WithRSAEncryption
         91:72:ce:a1:f5:17:3b:07:f3:83:9d:d5:87:54:0b:1f:f7:5f:
         74:7a:8e:ad:20:e7:66:e3:0d:67:5e:f9:0a:3c:a0:05:d5:62:
         11:1f:40:cc:02:30:9d:a1:b3:a1:b6:f6:96:7d:47:18:d2:19:
         91:f0:cd:ec:c6:2f:39:fb:25:3d:5c:27:5e:c4:e8:d4:45:58:
         e2:50:75:ca:02:a3:27:3a:62:2a:12:e3:99:8a:de:82:b7:08:
         05:8d:8c:29:5b:3d:e5:0f:a9:d5:66:6d:78:34:91:e4:cc:06:
         48:af:f8:75:f9:bf:85:55:1d:b6:fb:df:96:eb:3f:d5:e5:85:
         de:7d:e0:05:13:65:24:ea:8b:b0:ac:29:87:5e:6c:73:eb:78:
         47:b8:6a:29:69:f3:ca:4b:f1:d4:67:93:cf:0b:8a:7a:97:b7:
         92:1c:53:42:98:37:f6:a4:53:64:39:91:fc:45:dd:b4:7b:62:
         17:30:07:5e:c1:ca:6b:ec:b8:c9:8a:91:80:21:25:e5:49:43:
         f5:ef:40:d6:37:5a:20:85:c5:5b:c8:27:4d:8a:4e:e5:cc:a3:
         f1:a9:4b:76:9f:6b:a1:aa:6e:bd:ff:73:f0:46:67:81:e1:23:
         47:1b:aa:3b:8f:02:78:16:00:42:28:c8:58:e4:3e:21:2b:3a:
         8c:07:a3:e0
```

Table 56: Get Certificate Data Response

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 03 1C	Length	796	Number of bytes in Packet

Bytes (Hex)	Purpose	Value	Notes
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 03 04	Length	772	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 02 F7	Length	759	Number of bytes in Payload
Payload			
F0	Start List Token		Begin results list

D2 ED 30 82 02 EA 30 82 01 D2 A0 03 02 01 02 02 14 4D 77 BE E6 9A C9 0D 67 7C 30 A2 16 97 69 68 00 38 C2 B3 7A 30 0D 06 09 2A 86 48 86 F7 0D 01 01 0B 05 00 30 1B 31 19 30 17 06 03 55 04 03 0C 10 44 65 76 69 63 65 20 56 65 6E 64 6F 72 20 43 41 30 20 17 0D 32 31 31 31 33 30 31 35 32 34 35 33 5A 18 0F 32 30 35 32 31 31 32 32 31 35 32 34 35 33 5A 30 1A 31 18 30 16 06 03 55 04 03 0C 0F 4B 65 79 20 50 65 72 20 49 2F 4F 20 50 4B 49 30 82 01 22 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01 05 00 03 82 01 0F 00 30 82 01 0A 02 82 01 01 00 E9 53 56 10 6F D6 97 78 C2 C7 AA 62 8E 80 FB D0 B8 14 04 38 67 1F A0 7F 2E 81 94 D8 3B 0C E8 41 FA 0B 2B A7 90 B1 59 92 B4 98 80 EA 45 65 2C A4 ED B7 04 FA EF EE 5A DA 1F 1E 9D B6 D5 8F 3D 8C 90 35 7F 24 B6 1B 00 86 7C 3B 4C 37 7B A9 37 D6 79 AA 54 D1 AD DE D6 60 46 33 12 C9 91 B6 D3 02 88 CB D4 3A 7B 08 94 E1 A0 D9 66 1E 73 6D 11 4E 0E 87 48 37 A2 EA F2 2C DE 1C CD F0 F4 8B C5 83 BA 02 DE 06 B7 C5 39 E7 2B 8E B6 D5 EC 37 82 AE 29 4E 4D 12 E5 0F DE 5D BC 0D 4A B7 BD BD 27 79 2D FA 3A 4B BD AB E4 52 39 F3 84 BB BD 7C 3D 9C 6D 0A D4 C4 98 39 D3 5A A7 ED 7A 3C 26 DB AA CB 31 FD DD FF 72 D2 6F 33 8A B4 7A D8 BD 92 03 27 6F 52 2A 62 3B 0F 28 E7 EA 1A C2 4F 0F C4 3C 76 85 D7 60 E0 AA 7C CC 31 D0 EF A4 E6 19 F8 FA 03 8B 5C 01 F4 43 2C 8D 58 B7 41 93 FF 4B 5D E4 83 02 03 01 00 01 A3 25 30	Medium Atom Token		Certificate data in DER format from KeyPerIOPublicKeyCertificateData byte table. This data contained in this field is vendor unique.

Bytes (Hex)	Purpose	Value	Notes
23 30 13 06 03 55 1D 11 04 0C 30 0A 82 08 44 4E 53 20 4E 61 6D 65 30 0C 06 03 55 1D 13 01 01 FF 04 02 30 00 30 0D 06 09 2A 86 48 86 F7 0D 01 01 0B 05 00 03 82 01 01 00 91 72 CE A1 F5 17 3B 07 F3 83 9D D5 87 54 0B 1F F7 5F 74 7A 8E AD 20 E7 66 E3 0D 67 5E F9 0A 3C A0 05 D5 62 11 1F 40 CC 02 30 9D A1 B3 A1 B6 F6 96 7D 47 18 D2 19 91 F0 CD EC C6 2F 39 FB 25 3D 5C 27 5E C4 E8 D4 45 58 E2 50 75 CA 02 A3 27 3A 62 2A 12 E3 99 8A DE 82 B7 08 05 8D 8C 29 5B 3D E5 0F A9 D5 66 6D 78 34 91 E4 CC 06 48 AF F8 75 F9 BF 85 55 1D B6 FB DF 96 EB 3F D5 E5 85 DE 7D E0 05 13 65 24 EA 8B B0 AC 29 87 5E 6C 73 EB 78 47 B8 6A 29 69 F3 CA 4B F1 D4 67 93 CF 0B 8A 7A 97 B7 92 1C 53 42 98 37 F6 A4 53 64 39 91 FC 45 DD B4 7B 62 17 30 07 5E C1 CA 6B EC B8 C9 8A 91 80 21 25 E5 49 43 F5 EF 40 D6 37 5A 20 85 C5 5B C8 27 4D 8A 4E E5 CC A3 F1 A9 4B 76 9F 6B A1 AA 6E BD FF 73 F0 46 67 81 E1 23 47 1B AA 3B 8F 02 78 16 00 42 28 C8 58 E4 3E 21 2B 3A 8C 07 A3 E0			
F1	End List Token		End results list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

Bytes (Hex)	Purpose	Value	Notes
00	Pad	0	Included in ComPacket and Packet lengths

3.4.5.6 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.4.5.7 Inject KEK Wrapped with Public Key Request Message

This section describes the procedure a host follows to inject a wrapped KEK into a TPer, where the KEK is wrapped with the public key that was extracted using the method described in section 3.4.5.5.1. To indicate that the KEK is wrapped with the public key, the public key needs to be specified in the UniqueIdentifier field in the EncryptionKeyInformation structure for the KMIP Request Message. In the case of a wrapping KEK, the UniqueIdentifier field is set to the KMIP KEK UID. The public key does not have a KMIP UID that is tracked by the Key Per I/O SP. Instead, the public key is specified by setting the UniqueIdentifier field to the value in the Name column of the Certificate Table that is associated with the public key. In this example, the value of the Name column is Certificate1 (see section 3.4.5.4).

Security Protocol 0x03 Compacket Header (see Table 57 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
0010    00 00 02 c8 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 57: Inject KEK Wrapped with Public Key Compacket Header

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 01	ComID	0x0801	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 02 C8	Length	712	Number of bytes in KMIP Payload

Security Protocol 0x03 Compacket Payload - TTLV byte encoding of the Inject KEK wrapped with the Public Key Request Message (see Table 58 for a description of the following byte encoding):

```
0000   42 00 78 01 00 00 02 c0  42 00 77 01 00 00 00 38
0010   42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020   00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
```

```
0030   00 00 00 01 00 00 00 00  42 00 0d 02 00 00 00 04
0040   00 00 00 01 00 00 00 00  42 00 0f 01 00 00 02 78
0050   42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
0060   42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0070   42 00 79 01 00 00 02 50  42 00 94 07 00 00 00 24
0080   63 32 30 33 65 62 39 65  2d 30 63 39 64 2d 34 38
0090   37 66 2d 38 36 33 66 2d  38 30 36 38 33 62 32 66
00a0   36 65 34 61 00 00 00 00  42 00 57 05 00 00 00 04
00b0   00 00 00 02 00 00 00 00  42 01 25 01 00 00 00 70
00c0   42 00 2b 01 00 00 00 30  42 00 83 05 00 00 00 04
00d0   00 00 00 0b 00 00 00 00  42 00 28 05 00 00 00 04
00e0   00 00 00 03 00 00 00 00  42 00 2a 02 00 00 00 04
00f0   00 00 01 00 00 00 00 00  42 00 08 01 00 00 00 30
0100   42 00 9d 07 00 00 00 07  54 43 47 2d 53 57 47 00
0110   42 00 0a 07 00 00 00 03  55 49 44 00 00 00 00 00
0120   42 00 0b 08 00 00 00 08  00 00 12 02 00 01 00 01
0130   42 00 8f 01 00 00 01 90  42 00 40 01 00 00 01 88
0140   42 00 42 05 00 00 00 04  00 00 00 01 00 00 00 00
0150   42 00 45 08 00 00 01 00  a0 42 51 1f 38 fa de 27
0160   44 fa 01 bc b3 db 3a f0  3c e1 88 21 e8 9c c1 b4
0170   1a f2 df 59 af a1 19 81  57 c1 cf e6 cf 70 e9 12
0180   7c 05 82 23 b1 f0 d6 80  fc 10 35 a0 b0 d2 a6 2f
0190   ce 85 58 b3 c1 f0 93 7d  92 2b 05 6a ad 22 b0 cd
01a0   55 72 c9 e8 31 d3 bb f7  5a c0 5a 6a 49 65 ad e3
01b0   0b 2d a1 f0 4f af 25 d3  64 15 00 a3 17 46 d8 a9
01c0   5a f2 c8 7a 00 0f 57 fe  40 19 86 b5 9c 45 50 18
01d0   e8 10 ac e1 dd 13 de 63  c3 bc 42 41 1d a2 2d 6e
01e0   f3 ad 0e 96 29 4c 65 af  65 7b 06 91 26 5c 64 78
01f0   bf 20 c2 60 b9 1d eb 70  d1 82 61 21 2e a3 7f 44
0200   8b 38 09 5c 26 d9 5f cd  4f a3 d5 d5 f1 44 ed 26
0210   1b 0d 38 25 91 8d d6 25  b6 ed 18 b1 4e 72 8a 61
0220   18 7d e0 71 cb a9 f4 3a  4b f4 7d 2e 01 7e 72 95
0230   89 74 58 f6 aa a6 74 d1  5e 51 a1 10 ee 1a 76 e6
0240   17 54 11 fe 80 78 26 9d  69 88 12 1e f7 5e e4 cd
0250   ad 7f e4 3f 8f 11 38 9d  42 00 46 01 00 00 00 68
0260   42 00 9e 05 00 00 00 04  00 00 00 01 00 00 00 00
0270   42 00 36 01 00 00 00 50  42 00 94 07 00 00 00 0c
0280   43 65 72 74 69 66 69 63  61 74 65 31 00 00 00 00
0290   42 00 2b 01 00 00 00 30  42 00 28 05 00 00 00 04
02a0   00 00 00 04 00 00 00 00  42 00 5f 05 00 00 00 04
02b0   00 00 00 02 00 00 00 00  42 01 02 05 00 00 00 04
02c0   00 00 00 06 00 00 00 00  00 00 00 00 00 00 00 00
02d0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
03f0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol `0x03` Compacket Payload - XML encoding of Inject KEK wrapped with the Public Key Request Message:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<KMIP>
    <!-- Inject KEK wrapped with Public Key into TPer Through KMIP -->
    <RequestMessage>
        <!-- Request Header -->
        <RequestHeader>
```

```
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
            </ProtocolVersion>
            <!-- Note: BatchErrorContinuationOption shall not be present if BatchCount <= 1 -
->
            <!-- Note: BatchOrderOption is optional if BatchCount == 1. BatchOrderOption is
True by default -->
            <!-- Note: BatchCount required -->
            <BatchCount type="Integer" value="1"/>
        </RequestHeader>
        <!-- Batch Item 1: Inject Wrapped KEK -->
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- Note: UniqueBatchItemID required if BatchCount > 1 -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <RequestPayload>
                <!-- Note: Globally unique KMIP Key UID associated with KEK -->
                <UniqueIdentifier type="TextString" value="c203eb9e-0c9d-487f-863f-
80683b2f6e4a"/>
                <ObjectType type="Enumeration" value="SymmetricKey"/>
                <Attributes>
                    <!-- Note: Cryptographic parameters associated with KeyEncryptionKey1 -->
                    <CryptographicParameters>
                        <KeyRoleType type="Enumeration" value="KEK"/>
                        <CryptographicAlgorithm type="Enumeration" value="AES"/>
                        <CryptographicLength type="Integer" value="256"/>
                    </CryptographicParameters>
                    <!-- Note: TCG KEK UID from KeyEncryptionKey table associated with KEK --
>
                    <Attribute>
                        <VendorIdentification type="TextString" value="TCG-SWG"/>
                        <AttributeName type="TextString" value="UID"/>
                        <!-- KeyEncryptionKey1 UID -->
                        <AttributeValue type="ByteString" value="0000120200010001"/>
                    </Attribute>
                </Attributes>
                <!-- Note: Specify KEK key material -->
                <SymmetricKey>
                    <KeyBlock>
                        <KeyFormatType type="Enumeration" value="Raw"/>
                        <!-- Note: KeyValue is a ByteString since key is wrapped -->
                        <!-- KEK associated with KeyEncryptionKey1 is wrapped with Public Key
-->
```

```
                        <!-- Unwrapped new KEK =
000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F -->
                        <KeyValue type="ByteString"
value="a042511f38fade2744fa01bcb3db3af03ce18821e89cc1b41af2df59afa1198157c1cfe6cf70e9127c0582
23b1f0d680fc1035a0b0d2a62fce8558b3c1f0937d922b056aad22b0cd5572c9e831d3bbf75ac05a6a4965ade30b2
da1f04faf25d3641500a31746d8a95af2c87a000f57fe401986b59c455018e810ace1dd13de63c3bc42411da22d6e
f3ad0e96294c65af657b0691265c6478bf20c260b91deb70d18261212ea37f448b38095c26d95fcd4fa3d5d5f144e
d261b0d3825918dd625b6ed18b14e728a61187de071cba9f43a4bf47d2e017e7295897458f6aaa674d15e51a110ee
1a76e6175411fe8078269d6988121ef75ee4cdad7fe43f8f11389d"/>
                        <!-- Note: Specify public key used to wrap injected key -->
                        <KeyWrappingData>
                            <!-- Note: Method used to wrap Key Value -->
                            <WrappingMethod type="Enumeration" value="Encrypt"/>
                            <EncryptionKeyInformation>
                                <!-- Note: Certificate1 Name which contains information about
KeyPerIOPublicKeyCertificateData -->
                                <UniqueIdentifier type="TextString" value="Certificate1"/>
                                <CryptographicParameters>
                                    <!-- Note: Crypto Algorithm, padding method, Hashing
Algorithm, and MGF Hashing Algorithm used to wrap key -->
                                    <!-- Note: Mask Generator is not specified. The default
value is MGF1. -->
                                    <!-- Note: P Source is not specified. The default value
is an empty byte string. -->
                                    <CryptographicAlgorithm type="Enumeration" value="RSA"/>
                                    <PaddingMethod type="Enumeration" value="OAEP"/>
                                    <MaskGeneratorHashingAlgorithm type="Enumeration"
value="SHA-256"/>
                                </CryptographicParameters>
                            </EncryptionKeyInformation>
                        </KeyWrappingData>
                    </KeyBlock>
                </SymmetricKey>
            </RequestPayload>
        </BatchItem>
    </RequestMessage>
</KMIP>
```

Table 58: Inject KEK Wrapped with Public Key Request

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
RequestMessage	Structure		420078	01	0000 02c0	

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
RequestHeader	Structure		420077	01	0000 0038	
ProtocolVersion	Structure		420069	01	0000 0020	
ProtocolVersionMajor	Integer	2	42006A	02	0000 0004	00000002 00000000
ProtocolVersionMinor	Integer	1	42006B	02	0000 0004	00000001 00000000
BatchCount	Integer	1	42000D	02	0000 0004	00000001 00000000
BatchItem	Structure		42000F	01	0000 0278	
Operation	Enumeration	Import	42005C	05	0000 0004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	0000 0001	01 00000000000000
RequestPayload	Structure		420079	01	0000 0250	
UniqueIdentifier	TextString	c203eb9e-0c9d-487f-863f-80683b2f6e4a	420094	07	0000 0024	6332303365623965 2d306339642d3438 37662d383633662d 3830363833623266 36653461 00000000
ObjectType	Enumeration	Symmetric Key	420057	05	0000 0004	00000002 00000000
Attributes	Structure		420125	01	0000 0070	
CryptographicParameters	Structure		42002B	01	0000 0030	
KeyRoleType	Enumeration	KEK	420083	05	0000 0004	0000000b 00000000
CryptographicAlgorithm	Enumeration	AES	420028	05	0000 0004	00000003 00000000
CryptographicLength	Integer	256	42002A	02	0000 0004	00000100 00000000
Attribute	Structure		420008	01	0000 0030	
VendorIdentification	TextString	TCG-SWG	42009D	07	0000 0007	5443472d535747 00
AttributeName	TextString	UID	42000A	07	0000 0003	554944 0000000000

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
AttributeValue	ByteString	00001202 00010001	42000B	08	0000 0008	0000120200010001
SymmetricKey	Structure		42008F	01	0000 0190	
KeyBlock	Structure		420040	01	0000 0188	
KeyFormatType	Enumeration	Raw	420042	05	0000 0004	00000001 00000000

KeyValue	ByteString	a042511f3 8fade2744 fa01bcb3d b3af03ce1 8821e89c c1b41af2d f59afa119 8157c1cfe 6cf70e912 7c058223 b1f0d680f c1035a0b 0d2a62fce 8558b3c1f 0937d922 b056aad2 2b0cd557 2c9e831d 3bbf75ac0 5a6a4965 ade30b2d a1f04faf25 d3641500 a31746d8 a95af2c87 a000f57fe 401986b5 9c455018 e810ace1 dd13de63 c3bc4241 1da22d6ef 3ad0e962 94c65af65 7b069126 5c6478bf2 0c260b91 deb70d18 261212ea 37f448b38 095c26d9 5fcd4fa3d 5d5f144ed 261b0d38 25918dd6 25b6ed18 b14e728a 61187de0 71cba9f43 a4bf47d2e 017e7295 897458f6a aa674d15 e51a110e e1a76e61 75411fe80 78269d69 88121ef75 ee4cdad7f e43f8f113 89d	420045	08	0000 0100	a042511f38fade27 44fa01bcb3db3af0 3ce18821e89cc1b4 1af2df59afa11981 57c1cfe6cf70e912 7c058223b1f0d680 fc1035a0b0d2a62f ce8558b3c1f0937d 922b056aad22b0cd 5572c9e831d3bbf7 5ac05a6a4965ade3 0b2da1f04faf25d3 641500a31746d8a9 5af2c87a000f57fe 401986b59c455018 e810ace1dd13de63 c3bc42411da22d6e f3ad0e96294c65af 657b0691265c6478 bf20c260b91deb70 d18261212ea37f44 8b38095c26d95fcd 4fa3d5d5f144ed26 1b0d3825918dd625 b6ed18b14e728a61 187de071cba9f43a 4bf47d2e017e7295 897458f6aaa674d1 5e51a110ee1a76e6 175411fe8078269d 6988121ef75ee4cd ad7fe43f8f11389d

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
KeyWrappingData	Structure		420046	01	0000 0068	
WrappingMethod	Enumeration	Encrypt	42009E	05	0000 0004	00000001 00000000
EncryptionKeyInformation	Structure		420036	01	0000 0050	
UniqueIdentifier	TextString	Certificate 1	420094	07	0000 000c	4365727469666963 61746531 00000000
CryptographicParameters	Structure		42002B	01	0000 0030	
CryptographicAlgorithm	Enumeration	RSA	420028	05	0000 0004	00000004 00000000
PaddingMethod	Enumeration	OAEP	42005F	05	0000 0004	00000002 00000000
MaskGeneratorHashingAlgorithm	Enumeration	SHA-256	420102	05	0000 0004	00000006 00000000

3.4.5.7.1 Inject KEK Wrapped with Public Key Response Message

Security Protocol 0x03 Compacket Header (see Table 59 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 c8 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 59: Inject KEK Wrapped Response with Public Key

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 01	ComID	0x0801	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 C8	Length	200	Number of bytes in KMIP Payload

Security Protocol 0x03 Compacket Payload - TTLV byte encoding of Inject KEK wrapped with the Public Key Response Message:

```
0000    42 00 7b 01 00 00 00 c0  42 00 7a 01 00 00 00 48
0010    42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020    00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
0030    00 00 00 01 00 00 00 00  42 00 92 09 00 00 00 08
0040    00 00 00 00 00 00 00 00  42 00 0d 02 00 00 00 04
0050    00 00 00 01 00 00 00 00  42 00 0f 01 00 00 00 68
0060    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
0070    42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0080    42 00 7f 05 00 00 00 04  00 00 00 00 00 00 00 00
0090    42 00 7c 01 00 00 00 30  42 00 94 07 00 00 00 24
00a0    63 32 30 33 65 62 39 65  2d 30 63 39 64 2d 34 38
00b0    37 66 2d 38 36 33 66 2d  38 30 36 38 33 62 32 66
00c0    36 65 34 61 00 00 00 00  00 00 00 00 00 00 00 00
00d0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol `0x03` Compacket Payload - XML encoding of Inject KEK wrapped with the Public Key Response Message (see Table 60 for a description of the following byte encoding):

```
<KMIP>
    <!-- Inject KEK wrapped with Public Key Response -->
    <ResponseMessage>
        <ResponseHeader>
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
            </ProtocolVersion>
            <!-- TimeStamp is required -->
            <!-- 0 indicates real-time clock not supported (refer to TCG core 3.3.4.7.4) -->
            <TimeStamp type="DateTime" value="0"/>
            <BatchCount type="Integer" value="1"/>
        </ResponseHeader>
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- UniqueBatchItemID value shall match value in request -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <ResultStatus type="Enumeration" value="Success"/>
            <ResponsePayload>
                <!-- UniqueIdentifier value shall match value in request -->
                <UniqueIdentifier type="TextString" value="c203eb9e-0c9d-487f-863f-
80683b2f6e4a"/>
            </ResponsePayload>
        </BatchItem>
    </ResponseMessage>
</KMIP>
```

Table 60: Inject KEK Wrapped with Public Key Response

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
ResponseMessage	Structure		42007B	01	0000 00c0	
ResponseHeader	Structure		42007A	01	0000 0048	
ProtocolVersion	Structure		420069	01	0000 0020	
ProtocolVersionMajor	Integer	2	42006A	02	0000 0004	00000002 00000000
ProtocolVersionMinor	Integer	1	42006B	02	0000 0004	00000001 00000000
TimeStamp	DateTime	0	420092	09	0000 0008	00000000 00000000
BatchCount	Integer	1	42000D	02	0000 0004	00000001 00000000
BatchItem	Structure		42000F	01	0000 0068	
Operation	Enumeration	Import	42005C	05	0000 0004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	0000 0001	01 00000000000000
ResultStatus	Enumeration	Success	42007F	05	0000 0004	00000000 00000000
ResponsePayload	Structure		42007C	01	0000 0030	
UniqueIdentifier	TextString	c203eb9e-0c9d-487f-863f-80683b2f6e4a	420094	07	0000 0024	63323033656239652d306339642d343837662d383633662d383036383362326636653461 00000000

3.4.6 Inject XTS-AES-256 MEK Wrapped with KEK

The following subsections describe the procedure a host follows to inject an MEK wrapped with a KEK into a TPer. The procedure to inject the initial KEK is described in section 3.4.2.

3.4.6.1 Unlock Key Injection Interface

The procedure a host follows to unlock the Key Injection Interface (Security Protocol 0x3) is described in section 3.3.8.

3.4.6.2 Configure AllowedKeyEncryptionKeys to associate Wrapping KEK with Namespace

The procedure a host follows to associate a KEK, used to wrap a MEK (KeyEncryptionKey1), with a Namespace (KeyTagAllocation1) is described in section 3.3.10.

3.4.6.3 Inject XTS-AES-256 MEK Request Message

This section describes the procedure a host follows to inject a wrapped XTS-AES-256 MEK into a TPer through the key injection interface. In this example, the MEK is injected into the TPer's key cache and associated with NamespaceID=1 and KeyTag=1. Both MEK keys are wrapped with the same KEK, which is associated with KeyEncryptionKey1 and is currently residing within the TPer. The host specifies the KMIP Key UID "c51a6ce0-e11c-4320-80c2-f1f270d2368e" to reference the existing KEK associated with KeyEncryptionKey1, which was previously injected into the TPer (see section 3.4.2.2). The host assigns a new KMIP Key UID "dbf8d112-cd66-424a-a3e9-d5e1ae131fc7" for MEK Key1 and "7f3afd46-4bb0-4724-a1de-d5304f3b1301" for MEK Key2 (see Table 62).

Note: Both Key1 and Key2 must be injected into the TPer through the key injection interface in a single KMIP Request Message.

Security Protocol `0x03` Compacket Header (see Table 61 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
0010    00 00 04 c8 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 61: Inject MEK Request Compacket Header

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 01	ComID	0x0801	
00 00	ComIDExtension	0	
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 04 C8	Length	1224	Number of bytes in Packet

Security Protocol `0x03` Compacket Payload — TTLV byte encoding of the Inject MEK Request Message (see Table 62 for a description of the following byte encoding):

```
0000    42 00 78 01 00 00 04 c0  42 00 77 01 00 00 00 48
0010    42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020    00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
0030    00 00 00 01 00 00 00 00  42 00 10 06 00 00 00 08
0040    00 00 00 00 00 00 00 01  42 00 0d 02 00 00 00 04
0050    00 00 00 02 00 00 00 00  42 00 0f 01 00 00 02 30
0060    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
0070    42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0080    42 00 79 01 00 00 02 08  42 00 94 07 00 00 00 24
0090    64 62 66 38 64 31 31 32  2d 63 64 36 36 2d 34 32
```

```
00a0    34 61 2d 61 33 65 39 2d  64 35 65 31 61 65 31 33
00b0    31 66 63 37 00 00 00 00  42 00 57 05 00 00 00 04
00c0    00 00 00 02 00 00 00 00  42 01 25 01 00 00 00 f8
00d0    42 00 2b 01 00 00 00 30  42 00 83 05 00 00 00 04
00e0    00 00 00 03 00 00 00 00  42 00 28 05 00 00 00 04
00f0    00 00 00 03 00 00 00 00  42 00 2a 02 00 00 00 04
0100    00 00 01 00 00 00 00 00  42 00 08 01 00 00 00 38
0110    42 00 9d 07 00 00 00 07  54 43 47 2d 53 57 47 00
0120    42 00 0a 07 00 00 00 0b  4e 61 6d 65 73 70 61 63
0130    65 49 44 00 00 00 00 00  42 00 0b 02 00 00 00 04
0140    00 00 00 01 00 00 00 00  42 00 08 01 00 00 00 30
0150    42 00 9d 07 00 00 00 07  54 43 47 2d 53 57 47 00
0160    42 00 0a 07 00 00 00 06  4b 65 79 54 61 67 00 00
0170    42 00 0b 02 00 00 00 04  00 00 00 01 00 00 00 00
0180    42 00 4a 01 00 00 00 40  42 00 4b 05 00 00 00 04
0190    00 00 01 0b 00 00 00 00  42 00 4c 07 00 00 00 24
01a0    37 66 33 61 66 64 34 36  2d 34 62 62 30 2d 34 37
01b0    32 34 2d 61 31 64 65 2d  64 35 33 30 34 66 33 62
01c0    31 33 30 31 00 00 00 00  42 00 8f 01 00 00 00 c0
01d0    42 00 40 01 00 00 00 b8  42 00 42 05 00 00 00 04
01e0    00 00 00 01 00 00 00 00  42 00 45 08 00 00 00 28
01f0    7a 73 16 08 02 7d fc 59  12 19 36 ce 11 b4 34 b9
0200    01 ae 81 8a 7b 06 c6 18  13 4a 62 0e 43 c3 4c eb
0210    89 ef a7 34 e8 7e cd 8e  42 00 46 01 00 00 00 70
0220    42 00 9e 05 00 00 00 04  00 00 00 01 00 00 00 00
0230    42 00 36 01 00 00 00 58  42 00 94 07 00 00 00 24
0240    63 35 31 61 36 63 65 30  2d 65 31 31 63 2d 34 33
0250    32 30 2d 38 30 63 32 2d  66 31 66 32 37 30 64 32
0260    33 36 38 65 00 00 00 00  42 00 2b 01 00 00 00 20
0270    42 00 28 05 00 00 00 04  00 00 00 03 00 00 00 00
0280    42 00 11 05 00 00 00 04  00 00 00 0d 00 00 00 00
0290    42 00 0f 01 00 00 02 30  42 00 5c 05 00 00 00 04
02a0    00 00 00 2a 00 00 00 00  42 00 93 08 00 00 00 01
02b0    02 00 00 00 00 00 00 00  42 00 79 01 00 00 02 08
02c0    42 00 94 07 00 00 00 24  37 66 33 61 66 64 34 36
02d0    2d 34 62 62 30 2d 34 37  32 34 2d 61 31 64 65 2d
02e0    64 35 33 30 34 66 33 62  31 33 30 31 00 00 00 00
02f0    42 00 57 05 00 00 00 04  00 00 00 02 00 00 00 00
0300    42 01 25 01 00 00 00 f8  42 00 2b 01 00 00 00 30
0310    42 00 83 05 00 00 00 04  00 00 00 03 00 00 00 00
0320    42 00 28 05 00 00 00 04  00 00 00 03 00 00 00 00
0330    42 00 2a 02 00 00 00 04  00 00 01 00 00 00 00 00
0340    42 00 08 01 00 00 00 38  42 00 9d 07 00 00 00 07
0350    54 43 47 2d 53 57 47 00  42 00 0a 07 00 00 00 0b
0360    4e 61 6d 65 73 70 61 63  65 49 44 00 00 00 00 00
0370    42 00 0b 02 00 00 00 04  00 00 00 01 00 00 00 00
0380    42 00 08 01 00 00 00 30  42 00 9d 07 00 00 00 07
0390    54 43 47 2d 53 57 47 00  42 00 0a 07 00 00 00 06
03a0    4b 65 79 54 61 67 00 00  42 00 0b 02 00 00 00 04
03b0    00 00 00 01 00 00 00 00  42 00 4a 01 00 00 00 40
03c0    42 00 4b 05 00 00 00 04  00 00 01 0a 00 00 00 00
03d0    42 00 4c 07 00 00 00 24  64 62 66 38 64 31 31 32
03e0    2d 63 64 36 36 2d 34 32  34 61 2d 61 33 65 39 2d
03f0    64 35 65 31 61 65 31 33  31 66 63 37 00 00 00 00
0400    42 00 8f 01 00 00 00 c0  42 00 40 01 00 00 00 b8
0410    42 00 42 05 00 00 00 04  00 00 00 01 00 00 00 00
0420    42 00 45 08 00 00 00 28  28 c9 f4 04 c4 b8 10 f4
0430    cb cc b3 5c fb 87 f8 26  3f 57 86 e2 d8 0e d3 26
```

```
0440    cb c7 f0 e7 1a 99 f4 3b  fb 98 8b 9b 7a 02 dd 21
0450    42 00 46 01 00 00 00 70  42 00 9e 05 00 00 00 04
0460    00 00 00 01 00 00 00 00  42 00 36 01 00 00 00 58
0470    42 00 94 07 00 00 00 24  63 35 31 61 36 63 65 30
0480    2d 65 31 31 63 2d 34 33  32 30 2d 38 30 63 32 2d
0490    66 31 66 32 37 30 64 32  33 36 38 65 00 00 00 00
04a0    42 00 2b 01 00 00 00 20  42 00 28 05 00 00 00 04
04b0    00 00 00 03 00 00 00 00  42 00 11 05 00 00 00 04
04c0    00 00 00 0d 00 00 00 00  00 00 00 00 00 00 00 00
04d0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
05f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol `0x03` Compacket Payload - XML encoding of the Inject MEK Request Message:

```
<?xml version="1.0" encoding="UTF-8"?>
<KMIP>
    <!-- Inject XTS-AES-256 MEK into TPer NameSpace/KeyTag Through KMIP -->
    <RequestMessage>
        <!-- Request Header -->
        <RequestHeader>
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
            </ProtocolVersion>
            <!-- Note: BatchErrorContinuationOption is optional if BatchCount > 1.
BatchErrorContinuationOption is Stop by default -->
            <!-- Note: BatchOrderOption is required if BatchCount > 1 -->
            <BatchOrderOption type="Boolean" value="True"/>
            <!-- Note: BatchCount required -->
            <BatchCount type="Integer" value="2"/>
        </RequestHeader>
        <!-- Batch Item 1: Inject XTS-AES-256 MEK Key1 -->
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- Note: UniqueBatchItemID required if BatchCount > 1 -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <RequestPayload>
                <!-- Note: Globally unique KMIP Key UID associated with MEK -->
                <UniqueIdentifier type="TextString" value="dbf8d112-cd66-424a-a3e9-
d5e1ae131fc7"/>
                <ObjectType type="Enumeration" value="SymmetricKey"/>
                <Attributes>
                    <!-- Note: Cryptographic parameters associated with MEK -->
                    <CryptographicParameters>
                        <KeyRoleType type="Enumeration" value="DEK"/>
```

```
                        <CryptographicAlgorithm type="Enumeration" value="AES"/>
                        <CryptographicLength type="Integer" value="256"/>
                    </CryptographicParameters>
                    <!-- Note: NamespaceID associated with row in KeyAllocationTable table --
>
                    <Attribute>
                        <VendorIdentification type="TextString" value="TCG-SWG"/>
                        <AttributeName type="TextString" value="NamespaceID"/>
                        <AttributeValue type="Integer" value="1"/>
                    </Attribute>
                    <!-- Note: Key Tag Slot associated with Namespace -->
                    <Attribute>
                        <VendorIdentification type="TextString" value="TCG-SWG"/>
                        <AttributeName type="TextString" value="KeyTag"/>
                        <AttributeValue type="Integer" value="1"/>
                    </Attribute>
                    <!-- Note: Specify link to MEK Key2 -->
                    <Link>
                        <LinkType type="Enumeration" value="NextLink"/>
                        <LinkedObjectIdentifier type="TextString" value="7f3afd46-4bb0-4724-
a1de-d5304f3b1301"/>
                    </Link>
                </Attributes>
                <SymmetricKey>
                    <KeyBlock>
                        <KeyFormatType type="Enumeration" value="Raw"/>
                        <!-- Note: KeyValue is a ByteString since key is wrapped -->
                        <!-- MEK is wrapped with KEK associated with KeyEncryptionKey1 -->
                        <!-- Unwrapped MEK =
A0112233445566778899AABBCCDDEEFF000102030405060708090A0B0C0D0E0F -->
                        <!-- Unwrapped KEK =
000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F -->
                        <KeyValue type="ByteString"
value="7a731608027dfc59121936ce11b434b901ae818a7b06c618134a620e43c34ceb89efa734e87ecd8e"/>
                        <!-- Note: Specify previously injected KEK used to wrap injected MEK
-->
                        <KeyWrappingData>
                            <!-- Note: Method used to wrap Key Value -->
                            <WrappingMethod type="Enumeration" value="Encrypt"/>
                            <EncryptionKeyInformation>
                                <!-- Note: KMIP Key UID of KEK used to wrap injected key -->
                                <!-- KMIP Key UID associated with KeyEncryptionKey1 -->
                                <UniqueIdentifier  type="TextString" value="c51a6ce0-e11c-
4320-80c2-f1f270d2368e"/>
                                <CryptographicParameters>
```

```
                                <CryptographicAlgorithm type="Enumeration" value="AES"/>
                                <!-- Note: Crypto Algorithm used to wrap key -->
                                <BlockCipherMode type="Enumeration" value="NISTKeyWrap"/>
                            </CryptographicParameters>
                        </EncryptionKeyInformation>
                    </KeyWrappingData>
                </KeyBlock>
            </SymmetricKey>
        </RequestPayload>
    </BatchItem>
    <!-- Batch Item 2: Inject XTS-AES-256 MEK Key2 -->
    <BatchItem>
        <Operation type="Enumeration" value="Import"/>
        <!-- Note: UniqueBatchItemID required if BatchCount > 1 -->
        <UniqueBatchItemID type="ByteString" value="02"/>
        <RequestPayload>
            <!-- Note: Globally unique KMIP Key UID associated with MEK -->
            <UniqueIdentifier type="TextString" value="7f3afd46-4bb0-4724-a1de-
d5304f3b1301"/>
            <ObjectType type="Enumeration" value="SymmetricKey"/>
            <Attributes>
               <!-- Note: Cryptographic parameters associated with MEK -->
                <CryptographicParameters>
                    <KeyRoleType type="Enumeration" value="DEK"/>
                    <CryptographicAlgorithm type="Enumeration" value="AES"/>
                    <CryptographicLength type="Integer" value="256"/>
                </CryptographicParameters>
                <!-- Note: NamespaceID associated with row in KeyAllocationTable table --
>
                <Attribute>
                    <VendorIdentification type="TextString" value="TCG-SWG"/>
                    <AttributeName type="TextString" value="NamespaceID"/>
                    <AttributeValue type="Integer" value="1"/>
                </Attribute>
                <!-- Note: Key Tag Slot associated with Namespace -->
                <Attribute>
                    <VendorIdentification type="TextString" value="TCG-SWG"/>
                    <AttributeName type="TextString" value="KeyTag"/>
                    <AttributeValue type="Integer" value="1"/>
                </Attribute>
                <!-- Note: Specify link to MEK Key1 -->
                <Link>
                    <LinkType type="Enumeration" value="PreviousLink"/>
                    <LinkedObjectIdentifier type="TextString" value="dbf8d112-cd66-424a-
a3e9-d5e1ae131fc7"/>
```

```
                    </Link>
                </Attributes>
                <SymmetricKey>
                    <KeyBlock>
                        <KeyFormatType type="Enumeration" value="Raw"/>
                        <!-- Note: KeyValue is a ByteString since key is wrapped -->
                        <!-- MEK is wrapped with KEK associated with KeyEncryptionKey1 -->
                        <!-- Unwrapped MEK =
00112233445566778899AABBCCDDEEFF000102030405060708090A0B0C0D0E0F -->
                        <!-- Unwrapped KEK =
000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F -->
                        <KeyValue type="ByteString"
value="28c9f404c4b810f4cbccb35cfb87f8263f5786e2d80ed326cbc7f0e71a99f43bfb988b9b7a02dd21"/>
                        <!-- Note: Specify previously injected KEK used to wrap injected MEK
-->
                        <KeyWrappingData>
                            <!-- Note: Method used to wrap Key Value -->
                            <WrappingMethod type="Enumeration" value="Encrypt"/>
                            <EncryptionKeyInformation>
                                <!-- Note: KMIP Key UID of KEK used to wrap injected key -->
                                <!-- KMIP Key UID associated with KeyEncryptionKey1 -->
                                <UniqueIdentifier  type="TextString" value="c51a6ce0-e11c-
4320-80c2-f1f270d2368e"/>
                                <CryptographicParameters>
                                    <CryptographicAlgorithm type="Enumeration" value="AES"/>
                                    <!-- Note: Crypto Algorithm used to wrap key -->
                                    <BlockCipherMode type="Enumeration" value="NISTKeyWrap"/>
                                </CryptographicParameters>
                            </EncryptionKeyInformation>
                        </KeyWrappingData>
                    </KeyBlock>
                </SymmetricKey>
            </RequestPayload>
        </BatchItem>
    </RequestMessage>
</KMIP>
```

Table 62: Inject XTS-AES-256 MEK

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
RequestMessage	Structure		420078	01	0000 04c0	

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
RequestHeader	Structure		420077	01	0000 0048	
ProtocolVersion	Structure		420069	01	0000 0020	
ProtocolVersionMajor	Integer	2	42006A	02	0000 0004	00000002 00000000
ProtocolVersionMinor	Integer	1	42006B	02	0000 0004	00000001 00000000
BatchOrderOption	Boolean	True	420010	06	0000 0008	00000000 00000001
BatchCount	Integer	2	42000D	02	0000 0004	00000002 00000000
BatchItem	Structure		42000F	01	0000 0230	
Operation	Enumeration	Import	42005C	05	0000 0004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	0000 0001	01 00000000000000
RequestPayload	Structure		420079	01	0000 0208	
UniqueIdentifier	TextString	dbf8d112-cd66-424a-a3e9-d5e1ae131fc7	420094	07	0000 0024	646266386431313 22d636436362d34 3234612d6133653 92d643565316165 313331666337 00000000
ObjectType	Enumeration	SymmetricKey	420057	05	0000 0004	00000002 00000000
Attributes	Structure		420125	01	0000 00f8	
CryptographicParameters	Structure		42002B	01	0000 0030	
KeyRoleType	Enumeration	DEK	420083	05	0000 0004	00000003 00000000
CryptographicAlgorithm	Enumeration	AES	420028	05	0000 0004	00000003 00000000
CryptographicLength	Integer	256	42002A	02	0000 0004	00000100 00000000
Attribute	Structure		420008	01	0000 0038	

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
VendorIdentification	TextString	TCG-SWG	42009D	07	0000 0007	5443472d535747 00
AttributeName	TextString	NamespaceID	42000A	07	0000 000b	4e616d657370616 3654944 0000000000
AttributeValue	Integer	1	42000B	02	0000 0004	00000001 00000000
Attribute	Structure		420008	01	0000 0030	
VendorIdentification	TextString	TCG-SWG	42009D	07	0000 0007	5443472d535747 00
AttributeName	TextString	KeyTag	42000A	07	0000 0006	4b6579546167 0000
AttributeValue	Integer	1	42000B	02	0000 0004	00000001 00000000
Link	Structure		42004A	01	0000 0040	
LinkType	Enumeration	NextLink	42004B	05	0000 0004	0000010b 00000000
LinkedObjectIdentifier	TextString	7f3afd46-4bb0-4724-a1de-d5304f3b1301	42004C	07	0000 0024	376633616664343 62d346262302d34 3732342d6131646 52d643533303466 336231333031 00000000
SymmetricKey	Structure		42008F	01	0000 00c0	
KeyBlock	Structure		420040	01	0000 00b8	
KeyFormatType	Enumeration	Raw	420042	05	0000 0004	00000001 00000000
KeyValue	ByteString	7a731608027dfc5 9121936ce11b434 b901ae818a7b06c 618134a620e43c3 4ceb89efa734e87 ecd8e	420045	08	0000 0028	7a731608027dfc5 9121936ce11b434 b901ae818a7b06c 618134a620e43c3 4ceb89efa734e87 ecd8e
KeyWrappingData	Structure		420046	01	0000 0070	
WrappingMethod	Enumeration	Encrypt	42009E	05	0000 0004	00000001 00000000
EncryptionKeyInformation	Structure		420036	01	0000 0058	

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
UniqueIdentifier	TextString	c51a6ce0-e11c-4320-80c2-f1f270d2368e	420094	07	0000 0024	633531613663653 02d653131632d34 3332302d3830633 22d663166323730 643233363865 00000000
CryptographicParameters	Structure		42002B	01	0000 0020	
CryptographicAlgorithm	Enumeration	AES	420028	05	0000 0004	00000003 00000000
BlockCipherMode	Enumeration	NISTKeyWrap	420011	05	0000 0004	0000000d 00000000
BatchItem	Structure		42000F	01	0000 0230	
Operation	Enumeration	Import	42005C	05	0000 0004	0000002a 00000000
UniqueBatchItemID	ByteString	02	420093	08	0000 0001	02 00000000000000
RequestPayload	Structure		420079	01	0000 0208	
UniqueIdentifier	TextString	7f3afd46-4bb0-4724-a1de-d5304f3b1301	420094	07	0000 0024	376633616664343 62d346262302d34 3732342d6131646 52d643533303466 336231333031 00000000
ObjectType	Enumeration	SymmetricKey	420057	05	0000 0004	00000002 00000000
Attributes	Structure		420125	01	0000 00f8	
CryptographicParameters	Structure		42002B	01	0000 0030	
KeyRoleType	Enumeration	DEK	420083	05	0000 0004	00000003 00000000
CryptographicAlgorithm	Enumeration	AES	420028	05	0000 0004	00000003 00000000
CryptographicLength	Integer	256	42002A	02	0000 0004	00000100 00000000
Attribute	Structure		420008	01	0000 0038	
VendorIdentification	TextString	TCG-SWG	42009D	07	0000 0007	5443472d535747 00

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
AttributeName	TextString	NamespaceID	42000A	07	0000 000b	4e616d6573706163654944 0000000000
AttributeValue	Integer	1	42000B	02	0000 0004	00000001 00000000
Attribute	Structure		420008	01	0000 0030	
VendorIdentification	TextString	TCG-SWG	42009D	07	0000 0007	5443472d535747 00
AttributeName	TextString	KeyTag	42000A	07	0000 0006	4b6579546167 0000
AttributeValue	Integer	1	42000B	02	0000 0004	00000001 00000000
Link	Structure		42004A	01	0000 0040	
LinkType	Enumeration	PreviousLink	42004B	05	0000 0004	0000010a 00000000
LinkedObjectIdentifier	TextString	dbf8d112-cd66-424a-a3e9-d5e1ae131fc7	42004C	07	0000 0024	646266386431313 22d636436362d34 3234612d6133653 92d643565316165 313331666337 00000000
SymmetricKey	Structure		42008F	01	0000 00c0	
KeyBlock	Structure		420040	01	0000 00b8	
KeyFormatType	Enumeration	Raw	420042	05	0000 0004	00000001 00000000
KeyValue	ByteString	28c9f404c4b810f4cbccb35cfb87f8263f5786e2d80ed326cbc7f0e71a99f43bfb988b9b7a02dd21	420045	08	0000 0028	28c9f404c4b810f4cbccb35cfb87f8263f5786e2d80ed326cbc7f0e71a99f43bfb988b9b7a02dd21
KeyWrappingData	Structure		420046	01	0000 0070	
WrappingMethod	Enumeration	Encrypt	42009E	05	0000 0004	00000001 00000000
EncryptionKeyInformation	Structure		420036	01	0000 0058	

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
UniqueIdentifier	TextString	c51a6ce0-e11c-4320-80c2-f1f270d2368e	420094	07	0000 0024	633531613663653 02d653131632d34 3332302d3830633 22d663166323730 643233363865 00000000
CryptographicParameters	Structure		42002B	01	0000 0020	
CryptographicAlgorithm	Enumeration	AES	420028	05	0000 0004	00000003 00000000
BlockCipherMode	Enumeration	NISTKeyWrap	420011	05	0000 0004	0000000d 00000000

3.4.6.3.1 Inject XTS-AES-256 MEK Response Message

Security Protocol 0x03 Compacket Header (see Table 63 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
0010    00 00 01 38 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 63: Inject MEK Response Compacket Header

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 01	ComID	0x0801	
00 00	ComIDExtension	0	
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 01 38	Length	312	Number of bytes in Packet

Security Protocol 0x03 Compacket Payload - TTLV byte encoding of the Inject MEK Response Message (see Table 64 for a description of the following byte encoding):

```
0000    42 00 7b 01 00 00 01 30  42 00 7a 01 00 00 00 48
0010    42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020    00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
0030    00 00 00 01 00 00 00 00  42 00 92 09 00 00 00 08
0040    00 00 00 00 00 00 00 00  42 00 0d 02 00 00 00 04
0050    00 00 00 02 00 00 00 00  42 00 0f 01 00 00 00 68
0060    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
```

```
0070    42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0080    42 00 7f 05 00 00 00 04  00 00 00 00 00 00 00 00
0090    42 00 7c 01 00 00 00 30  42 00 94 07 00 00 00 24
00a0    64 62 66 38 64 31 31 32  2d 63 64 36 36 2d 34 32
00b0    34 61 2d 61 33 65 39 2d  64 35 65 31 61 65 31 33
00c0    31 66 63 37 00 00 00 00  42 00 0f 01 00 00 00 68
00d0    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
00e0    42 00 93 08 00 00 00 01  02 00 00 00 00 00 00 00
00f0    42 00 7f 05 00 00 00 04  00 00 00 00 00 00 00 00
0100    42 00 7c 01 00 00 00 30  42 00 94 07 00 00 00 24
0110    37 66 33 61 66 64 34 36  2d 34 62 62 30 2d 34 37
0120    32 34 2d 61 31 64 65 2d  64 35 33 30 34 66 33 62
0130    31 33 30 31 00 00 00 00  00 00 00 00 00 00 00 00
0140    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol `0x03` Compacket Payload - XML encoding of the Inject MEK Response Message:

```
<KMIP>
    <!-- Inject MEK Response -->
    <ResponseMessage>
        <ResponseHeader>
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
            </ProtocolVersion>
            <!-- TimeStamp is required -->
            <!-- 0 indicates real-time clock not supported (refer to TCG core 3.3.4.7.4) -->
            <TimeStamp type="DateTime" value="0"/>
            <BatchCount type="Integer" value="2"/>
        </ResponseHeader>
        <!-- Batch Item 1: Response to Inject XTS-AES-256 MEK Key1 -->
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- UniqueBatchItemID value shall match value in request -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <ResultStatus type="Enumeration" value="Success"/>
            <ResponsePayload>
                <!-- UniqueIdentifier value shall match value in request -->
                <UniqueIdentifier type="TextString" value="dbf8d112-cd66-424a-a3e9-
d5e1ae131fc7"/>
            </ResponsePayload>
        </BatchItem>
        <!-- Batch Item 1: Response to Inject XTS-AES-256 MEK Key2 -->
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- UniqueBatchItemID value shall match value in request -->
```

```
            <UniqueBatchItemID type="ByteString" value="02"/>
            <ResultStatus type="Enumeration" value="Success"/>
            <ResponsePayload>
                <!-- UniqueIdentifier value shall match value in request -->
                <UniqueIdentifier type="TextString" value="7f3afd46-4bb0-4724-a1de-
d5304f3b1301"/>
            </ResponsePayload>
        </BatchItem>
    </ResponseMessage>
</KMIP>
```

Table 64: Inject MEK Response

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
ResponseMessage	Structure		42007B	01	00000130	
ResponseHeader	Structure		42007A	01	00000048	
ProtocolVersion	Structure		420069	01	00000020	
ProtocolVersionMajor	Integer	2	42006A	02	00000004	00000002 00000000
ProtocolVersionMinor	Integer	1	42006B	02	00000004	00000001 00000000
TimeStamp	DateTime	0	420092	09	00000008	00000000 00000000
BatchCount	Integer	2	42000D	02	00000004	00000002 00000000
BatchItem	Structure		42000F	01	00000068	
Operation	Enumeration	Import	42005C	05	00000004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	00000001	01 00000000000000
ResultStatus	Enumeration	Success	42007F	05	00000004	00000000 00000000
ResponsePayload	Structure		42007C	01	00000030	
UniqueIdentifier	TextString	dbf8d112-cd66-424a-a3e9-d5e1ae131fc7	420094	07	00000024	64626638643131322d636436362d343234612d613365392d6435653161653133316663337 00000000
BatchItem	Structure		42000F	01	00000068	
Operation	Enumeration	Import	42005C	05	00000004	0000002a 00000000
UniqueBatchItemID	ByteString	02	420093	08	00000001	02 00000000000000
ResultStatus	Enumeration	Success	42007F	05	00000004	00000000 00000000
ResponsePayload	Structure		42007C	01	00000030	

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
UniqueIdentifier	TextString	7f3afd46-4bb0-4724-a1de-d5304f3b1301	420094	07	00000024	37663361666434362 d346262302d343732 342d613164652d643 53330346633623133 3031 00000000

3.4.7 Inject KEK2 Wrapped with KEK1 and Nonce

The following subsections describe the procedure a host follows to inject a KEK wrapped with an existing KEK and Nonce into a TPer. The procedure to inject the initial KEK is described in section 3.4.2. The procedure to retrieve a Nonce from the TPer is described in the following subsections.

3.4.7.1 Unlock Key Injection Interface

The procedure a host follows to unlock the Key Injection Interface (Security Protocol 0x3) is described in section 3.3.8.

3.4.7.2 Open Session to Key Per I/O SP as Admin1

The procedure a host follows to open a Session to the Key Per I/O SP as the Admin1 authority is described in section 3.3.3.1.

3.4.7.3 Enable Replay Protection

This section describes the procedure a host follows to enable replay protection by setting the ReplayProtectionEnabled column in the KPIOPolicies Table to TRUE (see Table 65). When replay protection is enabled, all MEKs and KEKs injected into the TPer through the Key Injection Interface must be encrypted with a Nonce that can be retrieved from the TPer (see section 3.4.7.6).

Note: Replay protection is an optional feature that is exposed to the host through the Replay Protection Supported field from Level 0 Discovery. See section 3.3.2.1 for the process the host follows to retrieve the Replay Protection Supported field.

The following pseudo-code is the signature of the Set method used to configure the ReplayProtectionEnabled column of the KPIOPolicies Table:

```
session[TSN:HSN] -> KPIOPolicies_SingleRow_UID.Set[Values = [ReplayProtectionEnabled
= TRUE]]
```

Security Protocol 0x01 ComPacket Payload – TCG Method byte encoding of the invoking Set method on the KPIOPolicies single row (see Table 65 for a description of the following byte encoding):

```
0000    00 00 00 00 08 00 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 48 00 00 10 01  00 00 00 01 00 00 00 00
0020    00 00 00 00 00 00 00 00  00 00 00 30 00 00 00 00
0030    00 00 00 00 00 00 00 24  f8 a8 00 00 12 03 00 00
0040    00 01 a8 00 00 00 06 00  00 00 17 f0 f2 01 f0 f2
0050    03 01 f3 f1 f3 f1 f9 f0  00 00 00 f1 00 00 00 00
0060    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
```

```
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 65: Enable Replay Protection

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 00	ComID	0x0800	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 48	Length	72	Number of bytes in Packet
Packet			
00 00 10 01	TPerSN	0x1001	TPer Session Number
00 00 00 01	HostSN	1	Host Session Number
00 00 00 00	SeqNumber	0	
00 00	Reserved	0	
00 00	AckType	0	
00 00 00 00	Acknowledgement	0	
00 00 00 30	Length	48	Number of bytes in Subpacket
Subpacket			
00 00 00 00 00 00	Reserved	0	
00 00	Kind	0	Subpacket type = data subpacket
00 00 00 24	Length	36	Number of bytes in Payload
Payload			
F8	Call Token		Begin Set Method
A8 00 00 12 03 00 00 00 01	Short Atom Token	KPIO_Policies_UID	KPIOPolicies Row UID
A8 00 00 00 06 00 00 00 17	Short Atom Token	SetMethod_UID	Set Method UID
F0	Start List Token		Begin method parameter list
F2	Start Name Token		Begin Name-Value pair

Bytes (Hex)	Purpose	Value	Notes
01	Tiny Atom Token	1	Values
F0	Start List Token		Begin values
F2	Start Name Token		Begin Name-Value pair
03	Tiny Atom Token	3	ReplayProtectionEnabled Column
01	Tiny Atom Token	1	true
F3	End Name Token		End Name-Value pair
F1	End List Token		End values
F3	End Name Token		End Name-Value pair
F1	End List Token		End method parameter list
F9	End of Data Token		End method
F0	Start List Token		Begin Method Status list
00	Tiny Atom Token	0	Status Code
00	Tiny Atom Token	0	Reserved
00	Tiny Atom Token	0	Reserved
F1	End List Token		End Method Status list

The response to the above Set method request is a Set method response (see section 3.3.3.3).

3.4.7.4 Enable Wrapping KEK with Existing KEK

The procedure a host follows to enable wrapping KeyEncryptionKey2 with KeyEncryptionKey1 is described in section 3.4.4.3.

3.4.7.5 Close Session

The open Session to the Key Per I/O SP is no longer required and will be closed. The procedure to close a Session is described in section 3.3.3.4.

3.4.7.6 Retrieve Nonce Command

In the following example, the host issues the Get Nonce Command to the TPer and the response payload contains a 16-byte Nonce.

Note: The length of the Nonce is vendor unique. See section 3.3.2.1 for the procedure to discover the Nonce length.

Security Protocol 0x02 – Get Nonce Command:

```
IF_RECV(ProtocolID=2, ComID=0x0006, NSID=0x1)
```

Security Protocol 0x02 IF-RECV command response payload:

Table 66: Get Nonce Command Response

Byte	Value (Hex)	Description
0 to 15	01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16	Nonce value. This field is unique per request.

3.4.7.7 Inject Wrapped KEK with Nonce Request Message

This section describes the procedure a host follows to inject a wrapped KEK into a TPer, where the KEK is wrapped with a Nonce retrieved from the TPer (see section 3.4.7.6). In this example, the KEK injected into the TPer is associated with KeyEncryptionKey2 and the wrapping KEK, which is associated with KeyEncryptionKey1, is currently residing within the TPer. The host specifies the KMIP Key UID "c51a6ce0-e11c-4320-80c2-f1f270d2368e" to reference the existing KEK associated with KeyEncryptionKey1, which was previously injected into the TPer (see section 3.4.2.2). The host assigns a new KMIP Key UID " 36cd788b-982a-4388-8dc8-a0721d8ea2ab" for the KEK associated with KeyEncryptionKey2 (see Table 68).

Security Protocol 0x03 Compacket Header (see Table 67 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
0010    00 00 02 30 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 67: Inject Wrapped KEK with Nonce Compacket Header

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 01	ComID	0x0801	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 02 30	Length	560	Number of bytes in KMIP Payload

Security Protocol 0x03 Compacket Payload - TTLV byte encoding of Inject wrapped KEK with the Nonce Request Message (see Table 68 for a description of the following byte encoding):

```
0000    42 00 78 01 00 00 02 38  42 00 77 01 00 00 00 38
0010    42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020    00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
0030    00 00 00 01 00 00 00 00  42 00 0d 02 00 00 00 04
0040    00 00 00 01 00 00 00 00  42 00 0f 01 00 00 01 f0
```

```
0050    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
0060    42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0070    42 00 79 01 00 00 01 c8  42 00 94 07 00 00 00 24
0080    33 36 63 64 37 38 38 62  2d 39 38 32 61 2d 34 33
0090    38 38 2d 38 64 63 38 2d  61 30 37 32 31 64 38 65
00a0    61 32 61 62 00 00 00 00  42 00 57 05 00 00 00 04
00b0    00 00 00 02 00 00 00 00  42 01 25 01 00 00 00 70
00c0    42 00 2b 01 00 00 00 30  42 00 83 05 00 00 00 04
00d0    00 00 00 0b 00 00 00 00  42 00 28 05 00 00 00 04
00e0    00 00 00 03 00 00 00 00  42 00 2a 02 00 00 00 04
00f0    00 00 01 00 00 00 00 00  42 00 08 01 00 00 00 30
0100    42 00 9d 07 00 00 00 07  54 43 47 2d 53 57 47 00
0110    42 00 0a 07 00 00 00 03  55 49 44 00 00 00 00 00
0120    42 00 0b 08 00 00 00 08  00 00 12 02 00 01 00 02
0130    42 00 8f 01 00 00 00 f0  42 00 40 01 00 00 00 e8
0140    42 00 42 05 00 00 00 04  00 00 00 01 00 00 00 00
0150    42 00 45 08 00 00 00 30  e7 e8 c7 b2 c9 92 65 a0
0160    5d ee 8d 31 ab 8f 89 9b  e9 ae 84 81 5f cd 2d f5
0170    26 a7 b1 74 88 30 6b 40  2c cd c4 4f 5d 2c e4 bd
0180    b9 43 a3 7d aa 0a 1e c6  42 00 46 01 00 00 00 98
0190    42 00 9e 05 00 00 00 04  00 00 00 01 00 00 00 00
01a0    42 00 36 01 00 00 00 80  42 00 94 07 00 00 00 24
01b0    63 35 31 61 36 63 65 30  2d 65 31 31 63 2d 34 33
01c0    32 30 2d 38 30 63 32 2d  66 31 66 32 37 30 64 32
01d0    33 36 38 65 00 00 00 00  42 00 2b 01 00 00 00 30
01e0    42 00 28 05 00 00 00 04  00 00 00 03 00 00 00 00
01f0    42 00 11 05 00 00 00 04  00 00 00 09 00 00 00 00
0200    42 00 ce 02 00 00 00 04  00 00 00 10 00 00 00 00
0210    42 00 3d 08 00 00 00 0c  01 02 03 04 05 06 07 08
0220    09 10 11 12 00 00 00 00  42 00 ff 08 00 00 00 10
0230    e2 e3 5a c9 8e 18 f3 00  29 51 f2 1f 1c 1c b5 2e
0240    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
03f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol `0x03` Compacket Payload - XML encoding of Inject wrapped KEK with the Nonce Request Message:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<KMIP>
    <!-- Inject wrapped KEK with Nonce into TPer Through KMIP -->
    <RequestMessage>
        <!-- Request Header -->
        <RequestHeader>
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
            </ProtocolVersion>
            <!-- Note: BatchErrorContinuationOption shall not be present if BatchCount <= 1 -
->
```

```
            <!-- Note: BatchOrderOption is optional if BatchCount == 1. BatchOrderOption is
True by default -->
            <!-- Note: BatchCount required -->
            <BatchCount type="Integer" value="1"/>
        </RequestHeader>
        <!-- Batch Item 1: Inject Wrapped KEK -->
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- Note: UniqueBatchItemID required if BatchCount > 1 -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <RequestPayload>
                <!-- Note: Globally unique KMIP Key UID associated with KEK -->
                <UniqueIdentifier type="TextString" value="36cd788b-982a-4388-8dc8-
a0721d8ea2ab"/>
                <ObjectType type="Enumeration" value="SymmetricKey"/>
                <Attributes>
                    <!-- Note: Cryptographic parameters associated with KeyEncryptionKey2 --
>                     <CryptographicParameters>
                        <KeyRoleType type="Enumeration" value="KEK"/>
                        <CryptographicAlgorithm type="Enumeration" value="AES"/>
                        <CryptographicLength type="Integer" value="256"/>
                    </CryptographicParameters>
                    <!-- Note: TCG KEK UID from KeyEncryptionKey table associated with KEK --
>
                    <Attribute>
                        <VendorIdentification type="TextString" value="TCG-SWG"/>
                        <AttributeName type="TextString" value="UID"/>
                        <!-- KeyEncryptionKey2 UID -->
                        <AttributeValue type="ByteString" value="0000120200010002"/>
                    </Attribute>
                </Attributes>
                <!-- Note: Specify KEK key material -->
                <SymmetricKey>
                    <KeyBlock>
                        <KeyFormatType type="Enumeration" value="Raw"/>
                        <!-- Note: KeyValue is a ByteString since key is wrapped -->
                        <!-- KEK associated with KeyEncryptionKey2 is wrapped with existing
KEK associated with KeyEncryptionKey1 -->
                        <!-- Unwrapped new KEK =
DEF102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F -->
                        <!-- Existing wrapping KEK =
000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F -->
                        <!-- Nonce = 0102030405060708091011121314151 6 -->
                        <!-- IV = 010203040506070809101112 -->
```

```
                        <KeyValue type="ByteString"
value="e7e8c7b2c99265a05dee8d31ab8f899be9ae84815fcd2df526a7b17488306b402ccdc44f5d2ce4bdb943a3
7daa0a1ec6"/>
                        <!-- Note: Specify previously injected key use to wrap injected key -
->
                        <KeyWrappingData>
                            <!-- Note: Method used to wrap Key Value -->
                            <WrappingMethod type="Enumeration" value="Encrypt"/>
                            <EncryptionKeyInformation>
                                <!-- Note: KMIP Key UID of KEK used to wrap injected key -->
                                <!-- KMIP Key UID associated with KeyEncryptionKey1 -->
                                <UniqueIdentifier  type="TextString" value="c51a6ce0-e11c-
4320-80c2-f1f270d2368e"/>
                                <CryptographicParameters>
                                    <!-- Note: Crypto Algorithm used to wrap key -->
                                    <CryptographicAlgorithm type="Enumeration" value="AES"/>
                                    <BlockCipherMode type="Enumeration" value="GCM"/>
                                    <!-- Note: Tag Length of AuthenticatedEncryptionTag in
bytes. Required for GCM -->
                                    <TagLength type="Integer" value="16"/>
                                </CryptographicParameters>
                                <IVCounterNonce type="ByteString"
value="010203040506070809101112"/>
                            </EncryptionKeyInformation>
                        </KeyWrappingData>
                    </KeyBlock>
                </SymmetricKey>
                <AuthenticatedEncryptionTag type="ByteString"
value="e2e35ac98e18f3002951f21f1c1cb52e"/>
            </RequestPayload>
        </BatchItem>
    </RequestMessage>
</KMIP>
```

Table 68: Inject Wrapped KEK with Nonce Request Message

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
RequestMessage	Structure		420078	01	0000 0228	
RequestHeader	Structure		420077	01	0000 0038	
ProtocolVersion	Structure		420069	01	0000 0020	

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
ProtocolVersionMajor	Integer	2	42006A	02	0000 0004	00000002 00000000
ProtocolVersionMinor	Integer	1	42006B	02	0000 0004	00000001 00000000
BatchCount	Integer	1	42000D	02	0000 0004	00000001 00000000
BatchItem	Structure		42000F	01	0000 01e0	
Operation	Enumeration	Import	42005C	05	0000 0004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	0000 0001	01 00000000000000
RequestPayload	Structure		420079	01	0000 01b8	
UniqueIdentifier	TextString	36cd788b-982a-4388-8dc8-a0721d8ea2ab	420094	07	0000 0024	333663643738386 22d393832612d34 3338382d3864633 82d613037323164 386561326162 00000000
ObjectType	Enumeration	SymmetricKey	420057	05	0000 0004	00000002 00000000
Attributes	Structure		420125	01	0000 0070	
CryptographicParameters	Structure		42002B	01	0000 0030	
KeyRoleType	Enumeration	KEK	420083	05	0000 0004	0000000b 00000000
CryptographicAlgorithm	Enumeration	AES	420028	05	0000 0004	00000003 00000000
CryptographicLength	Integer	256	42002A	02	0000 0004	00000100 00000000
Attribute	Structure		420008	01	0000 0030	
VendorIdentification	TextString	TCG-SWG	42009D	07	0000 0007	5443472d535747 00
AttributeName	TextString	UID	42000A	07	0000 0003	554944 0000000000
AttributeValue	ByteString	000012020001000 2	42000B	08	0000 0008	000012020001000 2

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
SymmetricKey	Structure		42008F	01	0000 00e0	
KeyBlock	Structure		420040	01	0000 00d8	
KeyFormatType	Enumeration	Raw	420042	05	0000 0004	00000001 00000000
KeyValue	ByteString	e7e8c7b2c99265a05dee8d31ab8f899be9ae84815fcd2df526a7b17488306b402ccdc44f5d2ce4bdb943a37daa0a1ec6	420045	08	0000 0030	e7e8c7b2c99265a05dee8d31ab8f899be9ae84815fcd2df526a7b17488306b402ccdc44f5d2ce4bdb943a37daa0a1ec6
KeyWrappingData	Structure		420046	01	0000 0088	
WrappingMethod	Enumeration	Encrypt	42009E	05	0000 0004	00000001 00000000
EncryptionKeyInformation	Structure		420036	01	0000 0070	
UniqueIdentifier	TextString	c51a6ce0-e11c-4320-80c2-f1f270d2368e	420094	07	0000 0024	633531613663653 02d653131632d34 3332302d3830633 22d663166323730 643233363865 00000000
CryptographicParameters	Structure		42002B	01	0000 0038	
CryptographicAlgorithm	Enumeration	AES	420028	05	0000 0004	00000003 00000000
BlockCipherMode	Enumeration	GCM	420011	05	0000 0004	00000009 00000000
TagLength	Integer	16	4200CE	02	0000 0004	00000010 00000000
IVCounterNonce	ByteString	010203040506070809101112	42003D	08	0000 000c	010203040506070809101112 00000000
AuthenticatedEncryptionTag	ByteString	e2e35ac98e18f3002951f21f1c1cb52e	4200FF	08	0000 0010	e2e35ac98e18f3002951f21f1c1cb52e

3.4.7.7.1 Inject Wrapped KEK with Nonce Response Message

Security Protocol 0x03 Compacket Header (see Table 69 for a description of the following byte encoding):

```
0000    00 00 00 00 08 01 00 00  00 00 00 00 00 00 00 00
0010    00 00 00 c8 00 00 00 00  00 00 00 00 00 00 00 00
```

Table 69: Inject Wrapped KEK with Nonce Response Message

Bytes (Hex)	Purpose	Value	Notes
ComPacket			
00 00 00 00	Reserved	0	
08 01	ComID	0x0801	This field is vendor unique
00 00	ComIDExtension	0	This field is vendor unique
00 00 00 00	OutstandingData	0	
00 00 00 00	MinTransfer	0	
00 00 00 C8	Length	200	Number of bytes in KMIP Payload

Security Protocol 0x03 Compacket Payload - TTLV byte encoding of Inject wrapped KEK with the Nonce Response Message (see Table 70 for a description of the following byte encoding):

```
0000    42 00 7b 01 00 00 00 c0  42 00 7a 01 00 00 00 48
0010    42 00 69 01 00 00 00 20  42 00 6a 02 00 00 00 04
0020    00 00 00 02 00 00 00 00  42 00 6b 02 00 00 00 04
0030    00 00 00 01 00 00 00 00  42 00 92 09 00 00 00 08
0040    00 00 00 00 00 00 00 00  42 00 0d 02 00 00 00 04
0050    00 00 00 01 00 00 00 00  42 00 0f 01 00 00 00 68
0060    42 00 5c 05 00 00 00 04  00 00 00 2a 00 00 00 00
0070    42 00 93 08 00 00 00 01  01 00 00 00 00 00 00 00
0080    42 00 7f 05 00 00 00 04  00 00 00 00 00 00 00 00
0090    42 00 7c 01 00 00 00 30  42 00 94 07 00 00 00 24
00a0    33 36 63 64 37 38 38 62  2d 39 38 32 61 2d 34 33
00b0    38 38 2d 38 64 63 38 2d  61 30 37 32 31 64 38 65
00c0    61 32 61 62 00 00 00 00  00 00 00 00 00 00 00 00
00d0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
...
01f0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Security Protocol 0x03 Compacket Payload - XML encoding of Inject wrapped KEK with Nonce Response Message:

```
<KMIP>
    <!-- Inject wrapped KEK Response -->
    <ResponseMessage>
        <ResponseHeader>
            <!-- Note: Protocol version required -->
            <ProtocolVersion>
                <ProtocolVersionMajor type="Integer" value="2"/>
                <ProtocolVersionMinor type="Integer" value="1"/>
```

```
            </ProtocolVersion>
            <!-- TimeStamp is required -->
            <!-- 0 indicates real-time clock not supported (refer to TCG core 3.3.4.7.4) -->
            <TimeStamp type="DateTime" value="0"/>
            <BatchCount type="Integer" value="1"/>
        </ResponseHeader>
        <BatchItem>
            <Operation type="Enumeration" value="Import"/>
            <!-- UniqueBatchItemID value shall match value in request -->
            <UniqueBatchItemID type="ByteString" value="01"/>
            <ResultStatus type="Enumeration" value="Success"/>
            <ResponsePayload>
                <!-- UniqueIdentifier value shall match value in request -->
                <UniqueIdentifier type="TextString" value="36cd788b-982a-4388-8dc8-
a0721d8ea2ab"/>
            </ResponsePayload>
        </BatchItem>
    </ResponseMessage>
</KMIP>
```

Table 70: Inject Wrapped KEK with Nonce Response Message

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
ResponseMessage	Structure		42007B	01	000000c0	
ResponseHeader	Structure		42007A	01	00000048	
ProtocolVersion	Structure		420069	01	00000020	
ProtocolVersionMajor	Integer	2	42006A	02	00000004	00000002 00000000
ProtocolVersionMinor	Integer	1	42006B	02	00000004	00000001 00000000
TimeStamp	DateTime	0	420092	09	00000008	00000000 00000000
BatchCount	Integer	1	42000D	02	00000004	00000001 00000000
BatchItem	Structure		42000F	01	00000068	
Operation	Enumeration	Import	42005C	05	00000004	0000002a 00000000
UniqueBatchItemID	ByteString	01	420093	08	00000001	01 00000000000000
ResultStatus	Enumeration	Success	42007F	05	00000004	00000000 00000000

Tag	Type	Value	TTLV Tag	TTLV Type	TTLV Size	TTLV Value + Padding
ResponsePayload	Structure		42007C	01	00000030	
UniqueIdentifier	TextString	36cd788b-982a-4388-8dc8-a0721d8ea2ab	420094	07	00000024	3336636437383862 2d393832612d3433 38382d386463382d 6130373231643865 61326162 00000000

4 Key Per I/O Usage Examples

4.1 Overview

This section describes examples that demonstrate the side effects of Key Per I/O operations on the state of the Key Per I/O SP Tables and TPer key cache. These examples can be viewed as a state transition diagram, with each table representing a state, and the operation invocation representing a state transition from the preceding example (state) to the following example (state).

Each table in the following section contains the columns:

a) Namespace: identifies the Namespace associated with the row in the table;

b) Key Tag: lists the Key Tags associated with the Namespace;

c) Managed by Key Per I/O: identifies whether or not the Namespace associated with this row is managed by the Key Per I/O SP;

d) AllowedKeyEncryptionKey: identifies the value stored in the AllowedKeyEncryptionKey column of the `KeyTagAllocation` Table; and

e) Key Cache: represents the internal key cache managed by the TPer. This is an example of how the TPer could store MEK's in a key cache before and after Key Per I/O SP is Activated. The intention of this column is to show the side effects of an operation on an MEK stored in the TPer.

There are two types of MEKs used throughout the examples: Factory-MEKn and KPIO-MEKn, where n is the index of the MEK. Factory-MEKn are MEKs that are not injected through the procedures outlined in the Key Per I/O SSC and are therefore out of scope. They are used to represent MEKs and user data that may exist in a Namespace when the Namespace is not managed by Key Per I/O SP. KPIO-MEKn are MEKs injected through the Key Injection Interface supported by Key Per I/O SP.

The examples in this section represent a TPer with 7 Namespaces and 3 key tags are supported per Namespace.

4.2 Key Per I/O Operation Examples

4.2.1 Activate Method with KPIOSC = 1

`Activate` the Key Per I/O SP where the Key Per I/O capability applies to all Namespaces in the NVM subsystem (KPIOSC=1) [3]. After activating the Key Per I/O SP where KPIOSC=1, all existing user data is no longer accessible and all Namespaces are managed by the Key Per I/O SP.

Table 71 is the initial state where the Key Per I/O SP is in the Manufactured-Inactive state and user data is associated with each Namespace that is represented by Factory-MEKn in the Key Cache column.

Table 71: Key Per I/O SP in Manufactured-Inactive state

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	N/A	N/A	N/A	Factory-MEK1
NS2	N/A	N/A	N/A	Factory-MEK2
NS3	N/A	N/A	N/A	Factory-MEK3
NS4	N/A	N/A	N/A	Factory-MEK4
NS5	N/A	N/A	N/A	Factory-MEK5
NS6	N/A	N/A	N/A	Factory-MEK6
NS7	N/A	N/A	N/A	Factory-MEK7

Table 72 is the state after the Key Per I/O SP is activated where KPIOSC=1. The Managed column transitions to TRUE for each Namespace and all existing user data are rendered inaccessible, which is represented by the Factory-MEKn being removed from the Key Cache column.

Table 72: Key Per I/O SP Activated where KPIOSC=1

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE		
NS2	KT0 KT1 KT2	TRUE		
NS3	KT0 KT1 KT2	TRUE		
NS4	KT0 KT1 KT2	TRUE		
NS5	KT0 KT1 KT2	TRUE		
NS6	KT0 KT1 KT2	TRUE		
NS7	KT0 KT1 KT2	TRUE		

4.2.2 Activate Method with KPIOSC = 0

`Activate` the Key Per I/O SP where Key Per I/O capability does not apply to all Namespaces in the NVM subsystem. Activating the Key Per I/O SP when KPIOSC=0 will not result in the loss of user data.

Table 73 is the initial state where the Key Per I/O SP is in the Manufactured-Inactive state.

Table 73: Key Per I/O SP in Manufactured-Inactive State

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	N/A	N/A	N/A	Factory-MEK1
NS2	N/A	N/A	N/A	Factory-MEK2
NS3	N/A	N/A	N/A	Factory-MEK3
NS4	N/A	N/A	N/A	Factory-MEK4
NS5	N/A	N/A	N/A	Factory-MEK5
NS6	N/A	N/A	N/A	Factory-MEK6
NS7	N/A	N/A	N/A	Factory-MEK7

Table 74 is the state after `Activate` of the Key Per I/O SP where KPIOSC=0. This results in the Managed column remaining FALSE for each Namespace and user data remains accessible, which is represented by the Factory-MEKn remaining in the Key Cache column.

Table 74: Key Per I/O SP Activated where KPIOSC=0

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	N/A	FALSE	N/A	Factory-MEK1
NS2	N/A	FALSE	N/A	Factory-MEK2
NS3	N/A	FALSE	N/A	Factory-MEK3
NS4	N/A	FALSE	N/A	Factory-MEK4
NS5	N/A	FALSE	N/A	Factory-MEK5
NS6	N/A	FALSE	N/A	Factory-MEK6
NS7	N/A	FALSE	N/A	Factory-MEK7

4.2.3 KeyTagAllocation Table Managed column transition from 0 to 1

Transition the Managed column [3] of the `KeyTagAllocation` Table from 0 to 1 for a Namespace. This case is applicable only if KPIOSC=0. If KPIOSC=1, then the Managed column of the `KeyTagAllocation` Table is set to 1 and it is not modifiable.

Note: When the Managed column transitions from 0 to 1 for a Namespace, all user data in the Namespace is rendered irrecoverable [3], which is represented by the Factory-MEKn being removed from the Key Cache column associated with NS5.

Table 75 is the initial state of the Key Per I/O SP where the Key Per I/O SP is Activated and NS1, NS2, NS3, NS4 are managed by the Key Per I/O SP.

Table 75: Key Per I/O SP Activate with NS1-NS4 managed by Key Per I/O SP

Namespace	Key Tag	Managed by Key Per I/O SP	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 76 is the state after the `Set` method is invoked on the Managed column of the `KeyTagAllocation` Table associated with NamespaceID=NS5 where the Managed column is set to TRUE. This results in the Managed

column transitioning to TRUE for NS5 and all existing user data is rendered irrecoverable in this Namespace, which is represented by the Factory-MEK13 being removed from the Key Cache column associated with NS5.

Table 76: After Transition Managed column to TRUE for NS5

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	KT0 KT1 KT2	TRUE		
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.2.4 KeyTagAllocation Table Managed column transition from 1 to 0

Transition the Managed column of the `KeyTagAllocation` Table from 1 to 0. This case is applicable only if KPIOSC=0. If KPIOSC=1, then the Managed column of the `KeyTagAllocation` Table is set to 1 and the column is not modifiable.

Table 77 is the initial state where the Key Per I/O SP is Activated and NS1, NS2, NS3, NS4 are managed by the Key Per I/O SP.

Table 77: Key Per I/O SP Activate with NS1-NS4 managed by Key Per I/O SP

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 78 is the state after the `Set` method is invoked on `KeyTagAllocation` Table Managed column associated with NamespaceID=NS2 where Managed column is set to FALSE. This results in the Managed column transitioning to FALSE for NS2 and all existing user data is rendered irrecoverable in NS2, which is represented by KPIO-MEK4, KPIO-MEK5, and KPIO-MEK6 being removed from the Key Cache column associated with NS2.

Table 78: After Transition Managed column to FALSE for NS2

Namespace	Key Tag	Managed Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	N/A	FALSE	N/A	
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.2.5 Increase Key Tags Allocated for Namespace

The NumberOfKeyTags column of the `KeyTagAllocation` is set to the number of Key Tags allocated for a Namespace. This column can be set to a value no larger than Number of Key Tags Supported per Namespace field (see section 3.3.2.1). In this example, the number of Key Tags allocated for NS2 is increased to 4.

Table 79 is the initial state where NS1, NS2, NS3, NS4 are managed by Key Per I/O SP, 3 Key Tags are allocated per Namespace, and the Number of Key Tags Supported per Namespace is 4.

Table 79: 3 Key Tags Allocated Per Namespace

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 80 is the state after the `Set` method is invoked on the NumberOfKeyTags column of the `KeyTagAllocation` Table associated with NamespaceID=NS2 where NumberOfKeyTags column is set to 4. NS2 now has an empty slot in the TPer key cache associated with the KeyTag=KT3.

Table 80: After Increasing Key Tags Allocated for NS2 to 4

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2 KT3	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6 Empty
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.2.6 Decrease Key Tags Allocated for Namespace

The NumberOfKeyTags column of the `KeyTagAllocation` is set to the number of Key Tags allocated for a Namespace. In this example, the number of Key Tags allocated for NS2 is decreased to 2. The KeyTag=KT2 does not contain a MEK before the number of Key Tags is decreased, which is represented by “Empty” in Table 81, since the NumberOfKeyTags column value can only be decreased if reducing the number of Key Tags will not cause MEKs to be removed from the TPer key cache.

Table 81 is the initial state where NS1, NS2, NS3, NS4 are managed by Key Per I/O SP and 3 Key Tags are allocated per Namespace.

Table 81: 3 Key Tags Allocated Per Namespace

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 Empty
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 82 is the state after the `Set` method is invoked on the NumberOfKeyTags column of the `KeyTagAllocation` Table associated with NamespaceID=NS2 where NumberOfKeyTags column is set to 2. NS2 now has 2 Key Tags allocated:

Table 82: After Decreasing Key Tags Allocated for NS2 to 2

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.2.7 Inject KEK and Associate with Namespace

Inject the KEK into the TPer and associate the KEK with a Namespace so the MEK can be unwrapped by the KEK when injected into the TPer. Associate the KEK with the Namespace by setting the AllowedKeyEncryptionKey column of the `KeyTagAllocation` row associated with the Namespace.

Table 83 is the initial state where the Key Per I/O SP is Activated, NS1, NS2, NS3, NS4 are managed by Key Per I/O SP, and there are no KEKs associated with any Namespaces.

Table 83: Key Per I/O SP before KEK Injection

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE		
NS2	KT0 KT1 KT2	TRUE		

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS3	KT0 KT1 KT2	TRUE		
NS4	KT0 KT1 KT2	TRUE		
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 84 is the state after KEK1 is injected into the TPer (see section 3.4.2) and KEK1 is associated with NS2 (see section 3.3.10):

Table 84: Key Per I/O SP after KEK Injection and Association with NS2

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE		
NS2	KT0 KT1 KT2	TRUE	KEK1	
NS3	KT0 KT1 KT2	TRUE		
NS4	KT0 KT1 KT2	TRUE		
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.2.8 Inject MEK

In this section, an MEK is injected into the TPer's key cache. Before the MEK is injected, the KEK that wraps the MEK must already be injected into the TPer and the KEK must be associated with the Namespace that contains the KeyTag where the MEK will be injected (see section 3.2.8).

Table 85 is the initial state where NS1, NS2, NS3, NS4 are managed by Key Per I/O SP and KEK1 is associated with NS2.

Table 85: Key Per I/O SP where KEK1 associated with NS2

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE		

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS2	KT0 KT1 KT2	TRUE	KEK1	
NS3	KT0 KT1 KT2	TRUE		
NS4	KT0 KT1 KT2	TRUE		
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 86 is the state after KPIO-MEK1, which is wrapped with KEK1, is injected into the TPer. KPIO-MEK1 is injected into KeyTag=KT0 for NS2.

Table 86: Key Per I/O SP after KPIO-MEK1 Injected into TPer

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE		
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1
NS3	KT0 KT1 KT2	TRUE		
NS4	KT0 KT1 KT2	TRUE		
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.2.9 Clear Single MEK Command

In this example, a single MEK is cleared from the TPer's key cache associated with NS3 and KT1.

Table 87 is the initial state where NS1, NS2, NS3, NS4 are managed by Key Per I/O SP and MEKs are associated with all Key Tag slots in the TPer key cache.

Table 87: KPIO-MEKn injected into all KeyTag slots

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 88 is the state after KPIO-MEK8 is cleared from the TPer's key cache as a result of the Clear Single MEK Command with parameters NSID=NS3 and KeyTag=KT1.

Table 88: KPIO-MEK8 cleared from NS3 KT1 Key Tag slot

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 Empty KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.2.10 Clear All MEK Command with NSID not equal to FFFFFFFFh

In this example, all MEKs associated with NS2 are cleared from the TPer's key cache. The MEKs are cleared from the TPer's key cache through the Clear All MEK command with NSID= NS2 in the command.

Table 89 is the initial state where NS1, NS2, NS3, NS4 are managed by Key Per I/O SP and MEKs are associated with Key Tag slots in the TPer's key cache.

Table 89: KPIO-MEKn injected into all KeyTag slots

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 90 is the state after the Clear All MEK Command with parameters NSID=NS2 is invoked.

Table 90: KPIO-MEKn cleared from all NS2 Key Tag slots

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.2.11 Clear All MEK Command with NSID equal to FFFFFFFFh

In this example, all MEKs are cleared from the TPer's key cache for each Namespace managed by Key Per I/O SP (NS1-NS4). The MEKs are cleared from the TPer's key cache through the Clear All MEK command with NSID=FFFFFFFFh in the command.

Table 91 is the initial state where NS1, NS2, NS3, NS4 are managed by the Key Per I/O SP and the MEKs are associated with each Namespace and key tag in the TPer's key cache.

Table 91: KPIO-MEKn injected into all KeyTag slots

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 92 is the state after the Clear All MEK Command with parameters NSID=FFFFFFFFh is invoked.

Table 92: KPIO-MEKn cleared from all Key Tag slots associated with Key Per I/O SP

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	
NS2	KT0 KT1 KT2	TRUE	KEK1	
NS3	KT0 KT1 KT2	TRUE	KEK2	
NS4	KT0 KT1 KT2	TRUE	KEK1	
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.2.12 Revert Key Per I/O SP with KPIOSC = 1

The following section shows the effects of invoking the `Revert` method on the Key Per I/O SP where the Key Per I/O capability applies to all Namespaces in the NVM subsystem (KPIOSC=1). The `Revert` method will cause all user data associated with a Namespace managed by the Key Per I/O SP to be rendered irrecoverable. In this case, all Namespaces in the NVM subsystem are managed by the Key Per I/O SP. Therefore, reverting the Key Per I/O SP will cause all user data, for all Namespaces in the NVM subsystem, to be rendered irrecoverable.

Table 93 is the initial state where all Namespaces are managed by the Key Per I/O SP.

Table 93: All Namespaces Managed by Key Per I/O SP

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	KT0 KT1 KT2	TRUE	KEK3	KPIO-MEK13
NS6	KT0 KT1 KT2	TRUE	KEK4	KPIO-MEK14
NS7	KT0 KT1 KT2	TRUE	KEK5	KPIO-MEK15

Table 94 is the state after Reverting the Key Per I/O SP. All existing user data are irrecoverable, which is represented by the KPIO-MEKn being removed from the Key Cache column.

Table 94: After Revert Key Per I/O SP

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	N/A	N/A	N/A	
NS2	N/A	N/A	N/A	
NS3	N/A	N/A	N/A	
NS4	N/A	N/A	N/A	
NS5	N/A	N/A	N/A	
NS6	N/A	N/A	N/A	
NS7	N/A	N/A	N/A	

4.2.13 Revert Key Per I/O SP with KPIOSC = 0

The following section shows the effects of invoking the `Revert` method on the Key Per I/O SP where the Key Per I/O capability does not apply to all Namespaces in the NVM subsystem (KPIOSC=0). The Revert method will cause all user data associated with a Namespace managed by the Key Per I/O SP to be rendered irrecoverable. In this case, some Namespaces in the NVM subsystem may be managed by the Key Per I/O SP. Therefore, reverting the Key Per I/O SP will render irrecoverable user data associated with Namespaces that have been configured to be managed by the Key Per I/O SP.

Table 95 is the initial state where the Key Per I/O SP is Activated and NS1, NS2, NS3, NS4 are managed by the Key Per I/O SP.

Table 95: NS1-NS4 managed by Key Per I/O SP

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 96 is the state after Reverting the Key Per I/O SP. All existing user data associated with Namespaces managed by the Key Per I/O SP are irrecoverable, which is represented by the KPIO-MEKn being removed from the Key Cache column. Note that the Factory-MEKn were not removed from the Key Cache column because the Namespaces associated with these columns were not managed by the Key Per I/O SP.

Table 96: After Revert Key Per I/O SP

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	N/A	N/A	N/A	
NS2	N/A	N/A	N/A	
NS3	N/A	N/A	N/A	
NS4	N/A	N/A	N/A	
NS5	N/A	N/A	N/A	Factory-MEK13
NS6	N/A	N/A	N/A	Factory-MEK14
NS7	N/A	N/A	N/A	Factory-MEK15

4.3 Reset Examples

4.3.1 TCG Power Cycle Interaction with TPer Key Cache

All MEKs injected into a TPer through the Key Injection Interface must be stored in non-volatile media, such that a TCG Power Cycle reset will result in all KPIO-MEKs being cleared from the TPer's Key Cache. In this example, a TCG Power Cycle causes all KPIO-MEKs associated with Namespaces managed by Key Per I/O SP to be cleared from the TPer's key cache.

Note: The Factory-MEKn are outside the scope of the Key Per I/O SP and these MEKs may remain in the TPer's key cache after the TCG Power Cycle.

Table 97 is the initial state where NS1, NS2, NS3, NS4 are managed by the Key Per I/O SP and the KPIO-MEKn are associated with these namespaces. NS5, NS6, and NS7 are not managed by the Key Per I/O SP.

Table 97: TPer Key Cache before TCG Power Cycle

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 98 is the state after a TCG Power Cycle is completed. KPIO-MEKn are cleared from the TPer's key cache associated with NS1, NS2, NS3, and NS4.

Table 98: TPer Key Cache after TCG Power Cycle

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	
NS2	KT0 KT1 KT2	TRUE	KEK1	
NS3	KT0 KT1 KT2	TRUE	KEK2	
NS4	KT0 KT1 KT2	TRUE	KEK1	
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.3.2 TCG Hardware Reset Interactions with TPer Key Cache

TCG Hardware Reset of Storage Device does not cause KPIO-MEKs in the TPer's key cache to be cleared.

Table 99 is the initial state where NS1, NS2, NS3, NS4 are managed by the Key Per I/O SP and MEKs are associated with all Key Tag slots in the TPer's key cache.

Table 99: TPer Key Cache before TCG Hardware Reset

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 100 is the state after a TCG Hardware Reset is completed. All KPIO-MEKn remain in the TPer's key cache.

Table 100: TPer Key Cache after TCG Hardware Reset

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.3.3 TPER_RESET Interactions with TPer Key Cache

A TPER_RESET of a Storage Device does not cause KPIO-MEKs in the TPer's key cache to be cleared.

Table 101 is the initial state where NS1, NS2, NS3, NS4 are managed by the Key Per I/O SP and MEKs are associated with Key Tag slots in the TPer's key cache.

Table 101: TPer Key Cache before TPER_RESET

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

Table 102 is the state after a TPER_RESET is completed. All KPIO-MEKn remain in the TPer's key cache.

Table 102: TPer Key Cache after TPER_RESET

Namespace	Key Tag	Managed by Key Per I/O	AllowedKeyEncryptionKey	Key Cache
NS1	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK1 KPIO-MEK2 KPIO-MEK3
NS2	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK4 KPIO-MEK5 KPIO-MEK6
NS3	KT0 KT1 KT2	TRUE	KEK2	KPIO-MEK7 KPIO-MEK8 KPIO-MEK9
NS4	KT0 KT1 KT2	TRUE	KEK1	KPIO-MEK10 KPIO-MEK11 KPIO-MEK12
NS5	N/A	FALSE	N/A	Factory-MEK13
NS6	N/A	FALSE	N/A	Factory-MEK14
NS7	N/A	FALSE	N/A	Factory-MEK15

4.3.4 TCG Power Cycle Interaction with KEK Access Locked State

After a TCG Power Cycle, the AccessLocked column in the `KeyEncryptionKey` Table is set to TRUE for rows where the AccessLockEnabled column is configured to TRUE and the LockOnReset column of the `KeyEncryptionKey` row contains the value Power Cycle. Since the LockOnReset restrictions for the `KeyEncryptionKey` Table [3] specifies the column must contain the value Power Cycle, this last condition is always true. When a `KeyEncryptionKey` row is transitioned to the Access Locked state, both AccessLocked and AccessLockEnabled are set to TRUE for the row, and hence the key associated with the row cannot be used to unwrap injected MEKs and KEKs. The KEK must be transitioned to an access unlocked state before the KEK can

be used to unwrap an injected MEK or KEK. This is accomplished by opening a session as an Authority with access to set the AccessLocked or AccessLockEnabled column to FALSE for the row associated with the KEK (see section 3.2.7).

Table 103 is the initial state where the LockOnReset column contains Power Cycle for all KEKs and AccessLockEnabled is TRUE for KEK1, KEK2, and KEK3.

Table 103: KEK Access State before TCG Power Cycle

KeyEncryptionKey	LockOnReset	AccessLockEnabled	AccessLocked
KEK1	{Power Cycle and Programmatic}	TRUE	FALSE
KEK2	{Power Cycle and Hardware Reset}	TRUE	FALSE
KEK3	{Power Cycle}	TRUE	FALSE
KEK4	{Power Cycle and Programmatic}	FALSE	FALSE

Table 104 is the state after a TCG Power Cycle is completed. The AccessLocked column is transitioned to TRUE for KEK1, KEK2, and KEK3.

Table 104: KEK Access State after TCG Power Cycle

KeyEncryptionKey	LockOnReset	AccessLockEnabled	AccessLocked
KEK1	{Power Cycle and Programmatic}	TRUE	TRUE
KEK2	{Power Cycle and Hardware Reset}	TRUE	TRUE
KEK3	{Power Cycle}	TRUE	TRUE
KEK4	{Power Cycle and Programmatic}	FALSE	FALSE

4.3.5 TCG Hardware Reset Interactions with KEK Access Locked State

After a TCG Hardware Reset, the AccessLocked column in the `KeyEncryptionKey` Table is set to TRUE for rows where the AccessLockEnabled column is configured to TRUE and the LockOnReset column of the `KeyEncryptionKey` row contains the value Hardware Reset.

Table 105 is the initial state where the LockOnReset column contains Hardware Reset and the AccessLockEnabled column is TRUE for KEK2.

Table 105: KEK Access State before TCG Hardware Reset

KeyEncryptionKey	LockOnReset	AccessLockEnabled	AccessLocked
KEK1	{Power Cycle and Programmatic}	TRUE	FALSE
KEK2	{Power Cycle and Hardware Reset}	TRUE	FALSE
KEK3	{Power Cycle}	TRUE	FALSE
KEK4	{Power Cycle and Programmatic}	FALSE	FALSE

Table 106 is the state after a TCG Hardware Reset is completed. The AccessLocked column is transitioned to TRUE for KEK2.

Table 106: KEK Access State after TCG Hardware Reset

KeyEncryptionKey	LockOnReset	AccessLockEnabled	AccessLocked
KEK1	{Power Cycle and Programmatic}	TRUE	FALSE
KEK2	{Power Cycle and Hardware Reset}	TRUE	TRUE
KEK3	{Power Cycle}	TRUE	FALSE
KEK4	{Power Cycle and Programmatic}	FALSE	FALSE

4.3.6 TPER_RESET Interactions with KEK Access Locked State

After a TPER_RESET, the AccessLocked column in the `KeyEncryptionKey` Table is set to TRUE for rows where the AccessLockEnabled column is configured to TRUE and the LockOnReset column of the `KeyEncryptionKey` row contains the value Programmatic.

Table 107 is the initial state where the LockOnReset column contains Programmatic and the AccessLockEnabled column is TRUE for KEK1.

Table 107: KEK Access State before TPER_RESET

KeyEncryptionKey	LockOnReset	AccessLockEnabled	AccessLocked
KEK1	{Power Cycle and Programmatic}	TRUE	FALSE
KEK2	{Power Cycle and Hardware Reset}	TRUE	FALSE
KEK3	{Power Cycle}	TRUE	FALSE
KEK4	{Power Cycle and Programmatic}	FALSE	FALSE

Table 108 is the state after a TPER_RESET is completed. The AccessLocked column is transitioned to TRUE for KEK1. The AccessLocked column remains FALSE for KEK4 even though the LockOnReset column contains Programmatic because the AccessLockEnabled column is FALSE.

Table 108: KEK Access State after TPER_RESET

KeyEncryptionKey	LockOnReset	AccessLockEnabled	AccessLocked
KEK1	{Power Cycle and Programmatic}	TRUE	TRUE
KEK2	{Power Cycle and Hardware Reset}	TRUE	FALSE
KEK3	{Power Cycle}	TRUE	FALSE
KEK4	{Power Cycle and Programmatic}	FALSE	FALSE