STORAGE DEVELOPER CONFERENCE
SDC 23
BY Developers FOR Developers

# How to use an Encryption Key per I/O

Presented by

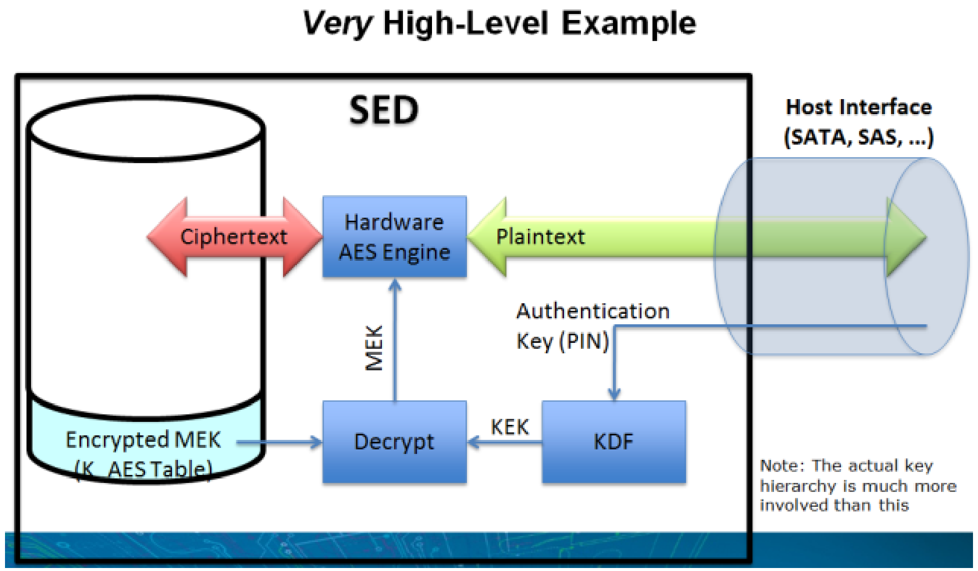
Eric Hibbard, CISSP, FIP, CISA
Samsung Semiconductor, Inc.

# Key per I/O (KPIO) Intro

Section Subtitle

# Background on Self Encrypting Drives (SEDs)

## Basic Data At Rest Protection Model:



## Properties:

- Encrypt all user accessible data all the time, at interface speeds
- Keys generated & stored in NVM by the storage device
- Media Encryption Key (MEK) associated with contiguous LBA ranges or Namespaces
- Opal/Enterprise SSC* deliver passwords to drive in the clear (when not using Trusted Computing Group (TCG)* - Secure Messaging)

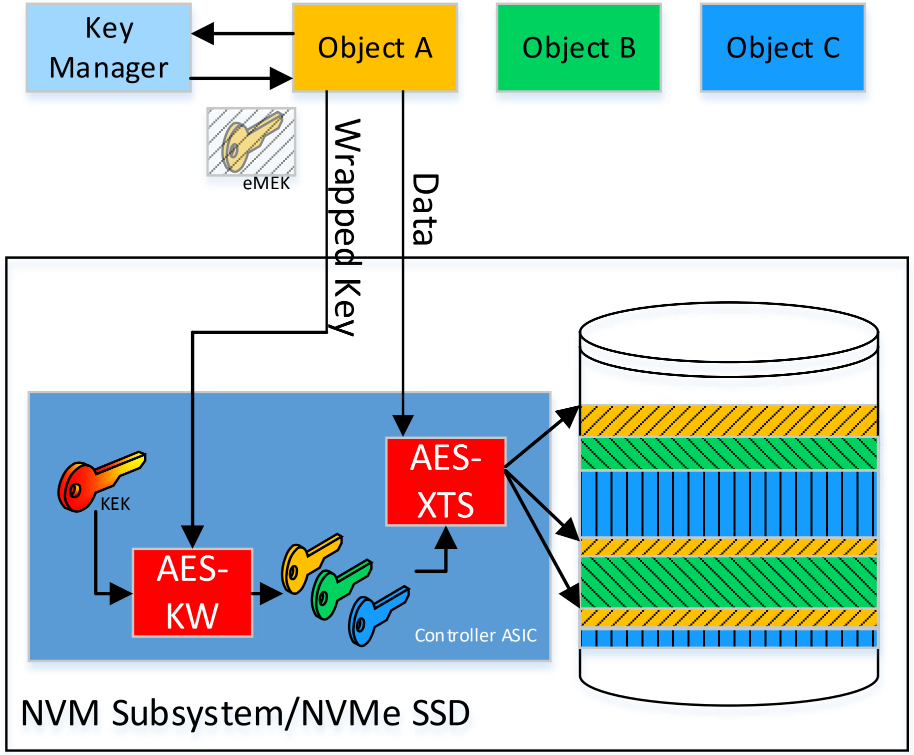
*Other names and brands may be claimed as the property of others.

# Key Per I/O

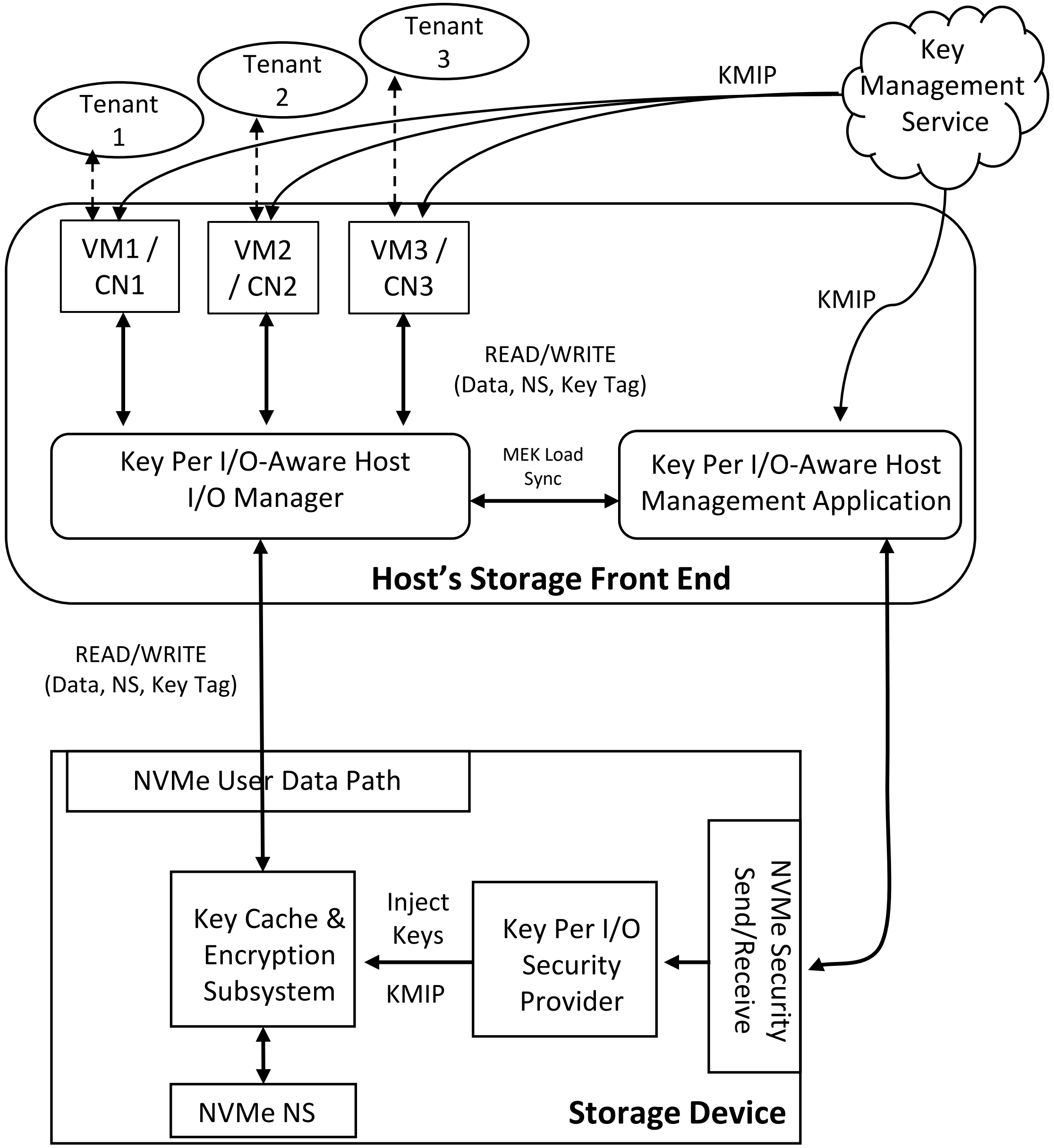
- Fine-grain data at rest encryption using storage devices (SSDs)
- Encryption engine in the storage device
- Key management controlled by the host
- Alignment with OASIS Key Management Interoperability Protocol (KMIP) Version 2.x

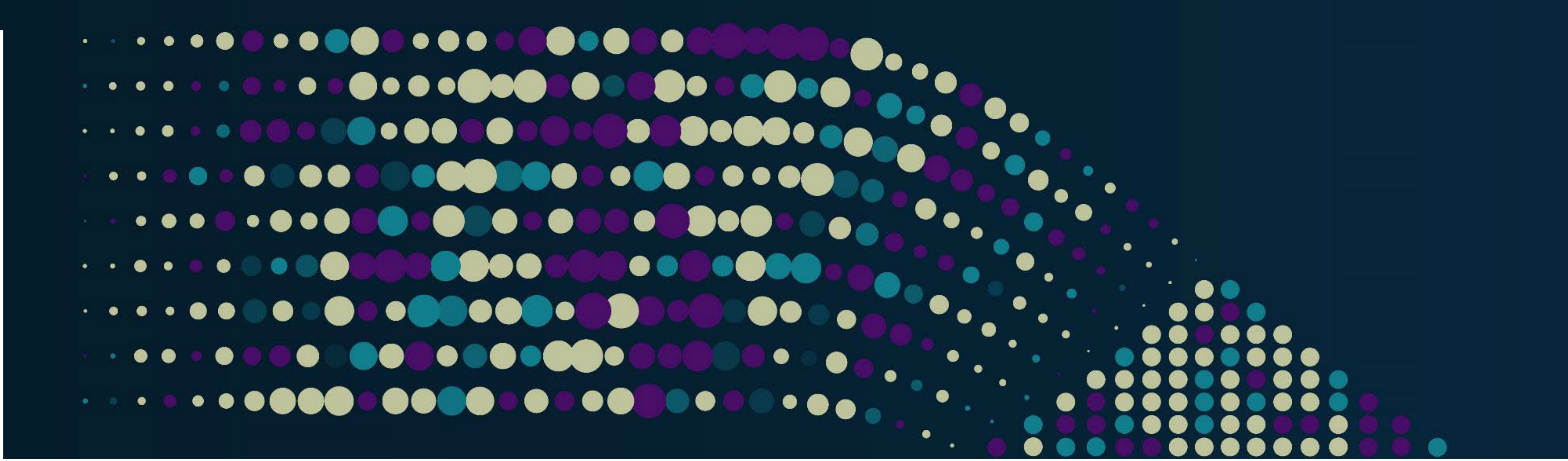
| Specification | Industry Standard Body | Status |
|---|---|---|
| NVMe® TP4055 | NVM Express | Ratified |
| TCG Key Per I/O SSC v1.00 | TCG | Published |
| TCG Key Per I/O Application Note v1.00 | TCG | In Public Review |
| TCG SIIS v1.11 | TCG | Published |
| TCG Key Per I/O Test Cases | TCG | Under Development |

# Key Per I/O Technology Overview



- Enables Storage Devices(SDs)' support of Host-Managed (i.e., Customer-managed) Storage Encryption Use Cases.
- Hosts no longer need to encrypt-at-compute with host/customer supplied encryption keys. They can now parallelize encryption across SDs with host-supplied Media Encryption Keys (MEKs) to increase storage systems' performance & bandwidth.
- Encrypted MEKs are injected into Self Encrypting Drive (SED)'s key cache and assigned a "Key Tag" by host software.
- Subsequent I/O can use the "Key Tag" to identify the MEK to encrypt/decrypt data to/from the SD in a non-contiguous fashion.
- MEKs are encrypted (wrapped) by a Key Encryption Key (KEK).
- KEKs may be supplied encrypted via RSA-based Key Wrapping.
- MEKs are not stored in the NVM of the drive and are lost on power loss.
- Cryptographic erase is done by deleting the MEK from the Key Manager and the SSD's key cache or by sanitizing entire SD.


Tenant 1
Tenant 2
Tenant 3
KMIP
Key Management Service
VM1 / CN1
VM2 / CN2
VM3 / CN3
KMIP
READ/WRITE (Data, NS, Key Tag)
Key Per I/O-Aware Host I/O Manager
MEK Load Sync
Key Per I/O-Aware Host Management Application
Host's Storage Front End
READ/WRITE (Data, NS, Key Tag)
NVMe User Data Path
Key Cache & Encryption Subsystem
Inject Keys
KMIP
Key Per I/O Security Provider
NVMe Security Send/Receive
NVMe NS
Storage Device



# Using Key Per I/O (KPIO)

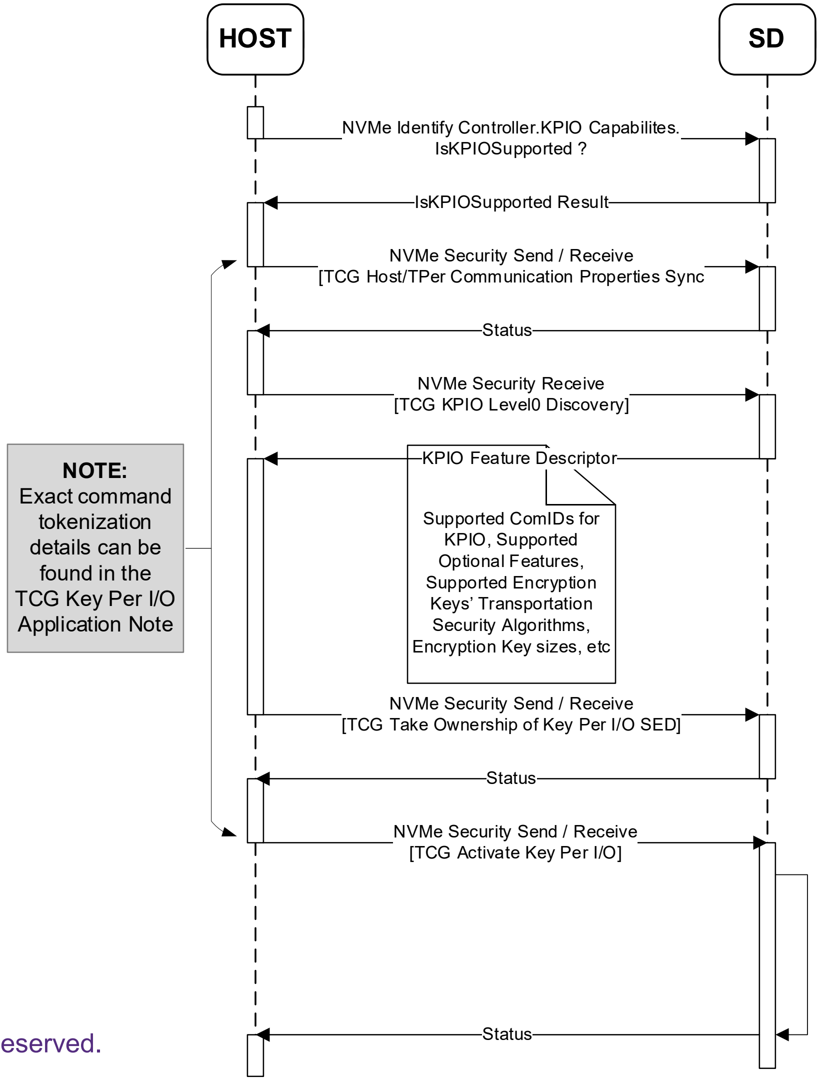
Section Subtitle

# Setting up KPIO (one time setup): Capabilities Discovery

- NVMe® Device Identify Discovery
  - Identify Controller
    - Key Per I/O Capabilities field
      - Key Per I/O Supported (KPIOS) bit
      - Key Per I/O Scope (KPIOSC) bit
  - Identify Namespace
    - Key Per I/O Status field
      - Key Per I/O Supported in Namespace (KPIONS) bit
      - Key Per I/O Enabled in Namespace (KPIOENS) bit
    - Maximum Key Tag (MAXKT) field
    - Key Per I/O Data Access Alignment and Granularity (KPIODAAG) field

- TCG Discovery (via NVMe® Security Receive)
  - Feature Level0 Discovery
    - Key Per I/O Security Protocols & ComIDs
    - Security properties for secure encryption key transport (RSA-OAEP wrapping, AES-GCM wrapping, etc..)
    - Number of Key Tags Supported (Globally vs Per-Namespace)
    - Maximum Supported Key Unique Identifier for Encryption Keys
    - Etc…
  - Namespace Level0 Discovery
    - Managed By Key Per I/O bit
    - Number of Allocated Key Tags

# Setting up KPIO (One Time Setup): Enabling KPIO



HOST

SD

NVMe Identify Controller.KPIO Capabilites.
IsKPIOSupported ?

IsKPIOSupported Result

NVMe Security Send / Receive
[TCG Host/TPer Communication Properties Sync

Status

NVMe Security Receive
[TCG KPIO Level0 Discovery]

KPIO Feature Descriptor

Supported ComIDs for KPIO, Supported Optional Features, Supported Encryption Keys' Transportation Security Algorithms, Encryption Key sizes, etc

NVMe Security Send / Receive
[TCG Take Ownership of Key Per I/O SED]

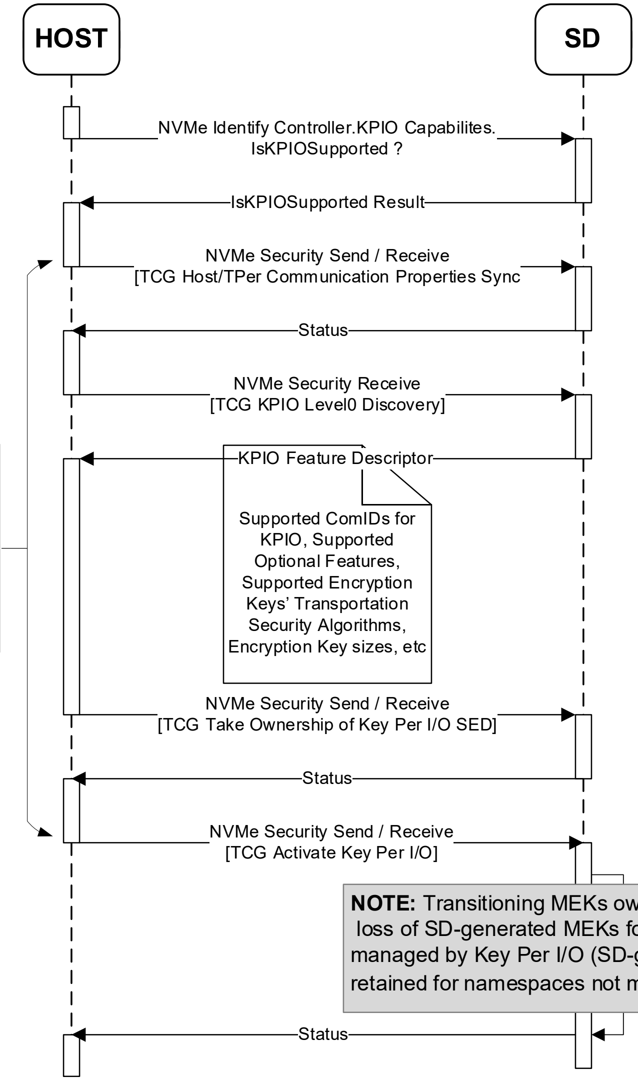
Status

NVMe Security Send / Receive
[TCG Activate Key Per I/O]

Status

**NOTE:**
Exact command tokenization details can be found in the TCG Key Per I/O Application Note

# Setting up KPIO (One Time Setup): Enabling KPIO



HOST

SD

NVMe Identify Controller.KPIO Capabilites.
IsKPIOSupported ?

IsKPIOSupported Result

NVMe Security Send / Receive
[TCG Host/TPer Communication Properties Sync

Status

NVMe Security Receive
[TCG KPIO Level0 Discovery]

KPIO Feature Descriptor

Supported ComIDs for KPIO, Supported Optional Features, Supported Encryption Keys' Transportation Security Algorithms, Encryption Key sizes, etc

NVMe Security Send / Receive
[TCG Take Ownership of Key Per I/O SED]

Status

NVMe Security Send / Receive
[TCG Activate Key Per I/O]

Status

**NOTE:** Exact command tokenization details can be found in the TCG Key Per I/O Application Note

**NOTE:** Transitioning MEKs ownership to the host implies loss of SD-generated MEKs for namespaces managed by Key Per I/O (SD-generated MEKs are retained for namespaces not managed by Key Per I/O).

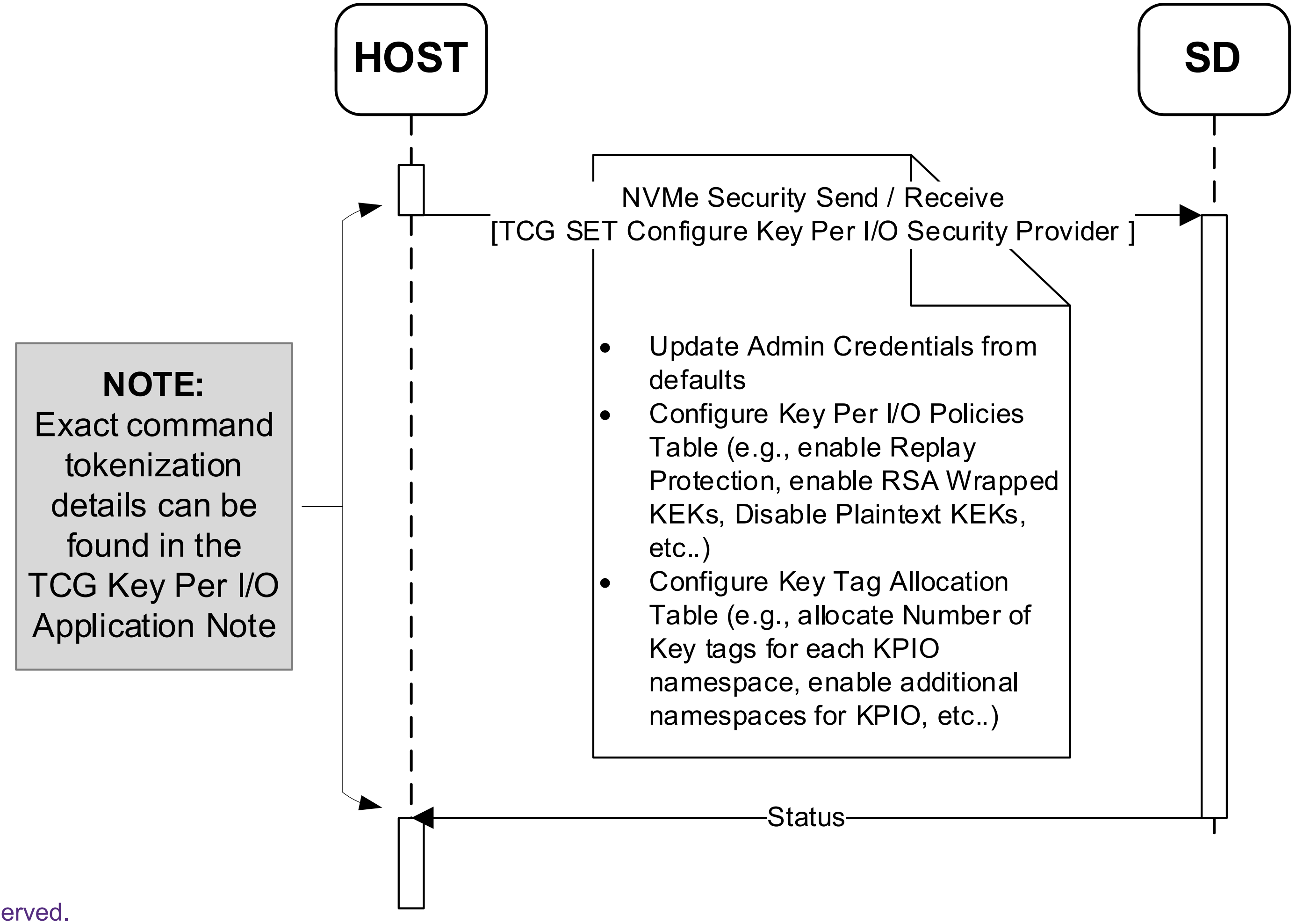
### SD's Volatile Key Cache/Table State On Cmd Completion if KPIO SCOPE is all of the NVM Subsystem (Example)

| | | |
|---|---|---|
| NSID0 | KeyTag0 | EMPTY |
| NSID1 | KeyTag0 | EMPTY |
| NSID1 | KeyTag1 | EMPTY |
| | … | |
| NSIDNN | KeyTagM | EMPTY |

### SD's Volatile Key Cache/Table State On Cmd Completion if KPIO SCOPE is per Namespace (Example)

| | | |
|---|---|---|
| NSID0 | N/A | SD's Factory Key0 |
| NSID1 | N/A | SD's Factory Key1 |
| | … | |
| NSIDNN | N/A | SD's Factory KeyNN |

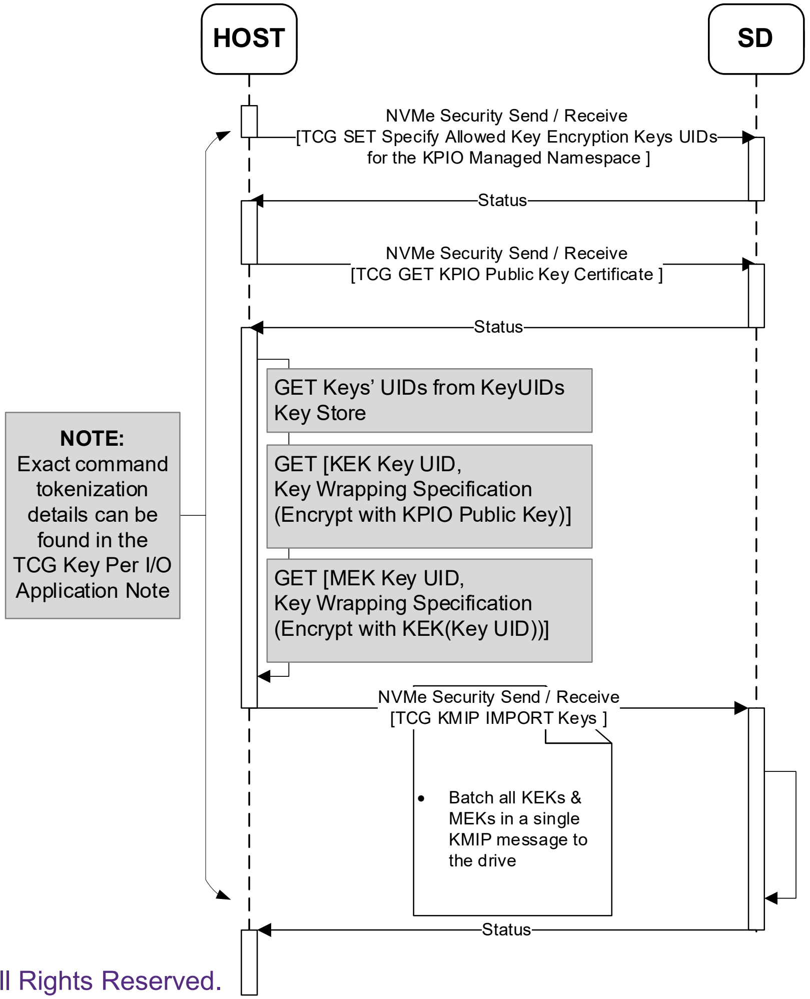
# Setting up KPIO (One Time Setup): Configuring KPIO


HOST
SD
NVMe Security Send / Receive
[TCG SET Configure Key Per I/O Security Provider ]
Status

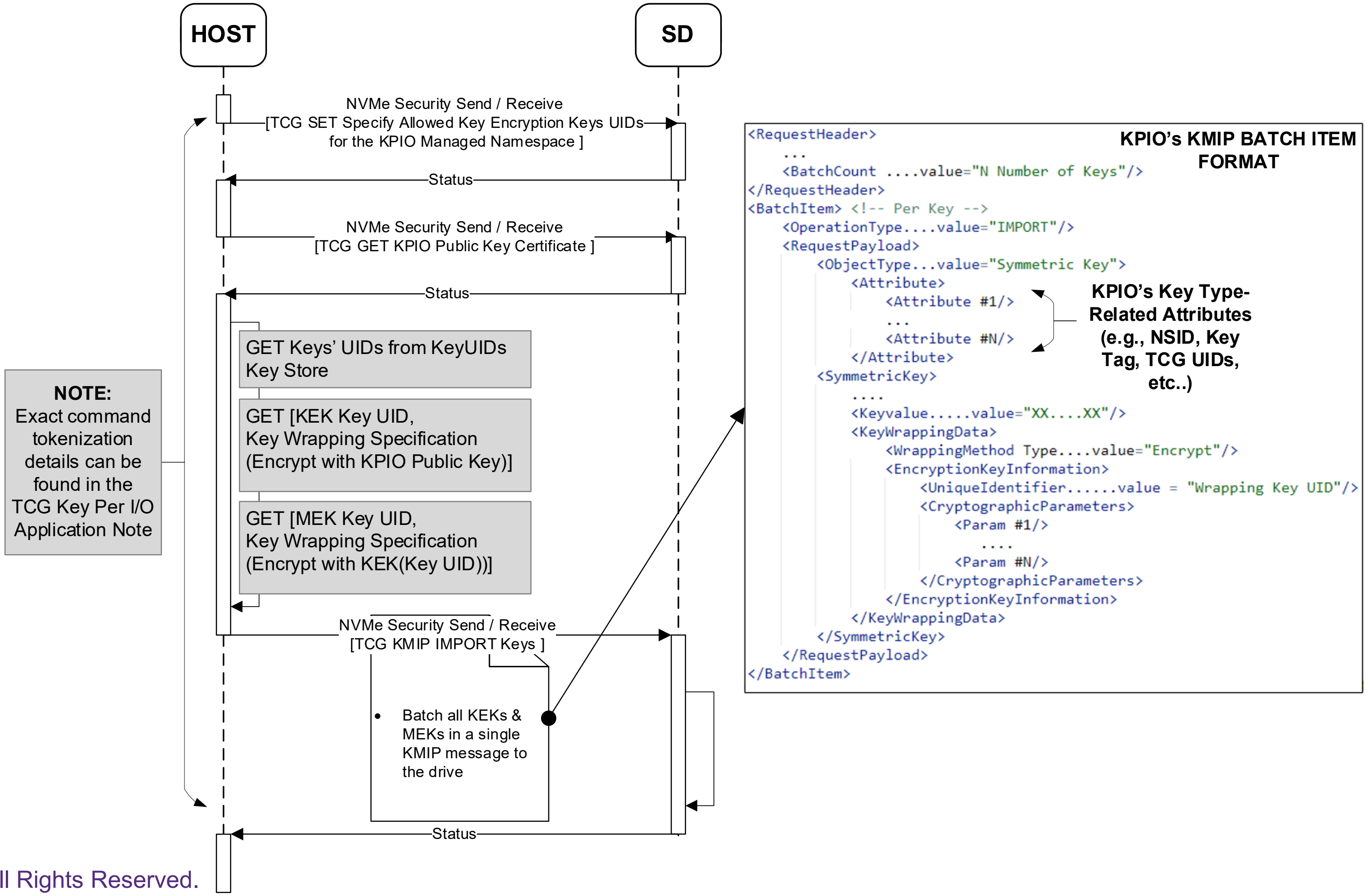
- Update Admin Credentials from defaults
- Configure Key Per I/O Policies Table (e.g., enable Replay Protection, enable RSA Wrapped KEKs, Disable Plaintext KEKs, etc..)
- Configure Key Tag Allocation Table (e.g., allocate Number of Key tags for each KPIO namespace, enable additional namespaces for KPIO, etc..)

**NOTE:**
Exact command tokenization details can be found in the TCG Key Per I/O Application Note

# Host Management of the SD’s Key Cache: Initial Loading of KEKs & MEKs


HOST
SD
NVMe Security Send / Receive
[TCG SET Specify Allowed Key Encryption Keys UIDs
for the KPIO Managed Namespace ]
Status
NVMe Security Send / Receive
[TCG GET KPIO Public Key Certificate ]
Status
GET Keys’ UIDs from KeyUIDs Key Store
GET [KEK Key UID, Key Wrapping Specification (Encrypt with KPIO Public Key)]
GET [MEK Key UID, Key Wrapping Specification (Encrypt with KEK(Key UID))]
NVMe Security Send / Receive
[TCG KMIP IMPORT Keys ]
• Batch all KEKs & MEKs in a single KMIP message to the drive
Status
NOTE:
Exact command tokenization details can be found in the TCG Key Per I/O Application Note

# Host Management of the SD’s Key Cache: Initial Loading of KEKs & MEKs


HOST
SD
NVMe Security Send / Receive
[TCG SET Specify Allowed Key Encryption Keys UIDs for the KPIO Managed Namespace ]
Status
NVMe Security Send / Receive
[TCG GET KPIO Public Key Certificate ]
Status
GET Keys’ UIDs from KeyUIDs Key Store
GET [KEK Key UID, Key Wrapping Specification (Encrypt with KPIO Public Key)]
GET [MEK Key UID, Key Wrapping Specification (Encrypt with KEK(Key UID))]
NVMe Security Send / Receive
[TCG KMIP IMPORT Keys ]
Status

- Batch all KEKs & MEKs in a single KMIP message to the drive

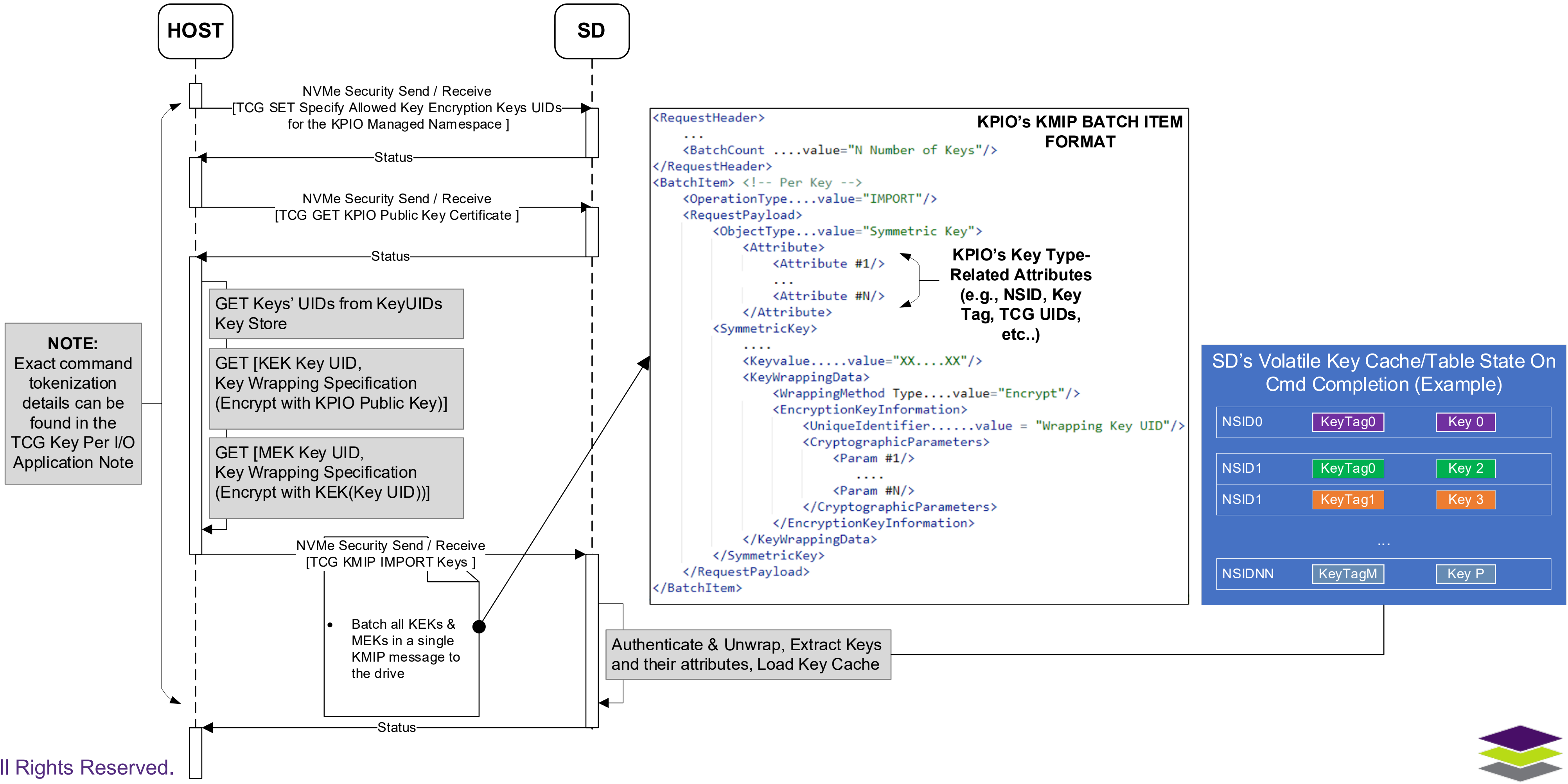
**NOTE:** Exact command tokenization details can be found in the TCG Key Per I/O Application Note

**KPIO’s KMIP BATCH ITEM FORMAT**

```
<RequestHeader>
    ...
    <BatchCount ....value="N Number of Keys"/>
</RequestHeader>
<BatchItem> <!-- Per Key -->
    <OperationType....value="IMPORT"/>
    <RequestPayload>
        <ObjectType...value="Symmetric Key">
            <Attribute>
                <Attribute #1/>
                ...
                <Attribute #N/>
            </Attribute>
        <SymmetricKey>
            ....
            <Keyvalue.....value="XX....XX"/>
            <KeyWrappingData>
                <WrappingMethod Type....value="Encrypt"/>
                <EncryptionKeyInformation>
                    <UniqueIdentifier......value = "Wrapping Key UID"/>
                    <CryptographicParameters>
                        <Param #1/>
                           ....
                        <Param #N/>
                    </CryptographicParameters>
                </EncryptionKeyInformation>
            </KeyWrappingData>
        </SymmetricKey>
    </RequestPayload>
</BatchItem>
```

**KPIO’s Key Type-Related Attributes (e.g., NSID, Key Tag, TCG UIDs, etc..)**

# Host Management of the SD’s Key Cache: Initial Loading of KEKs & MEKs



HOST

SD

NVMe Security Send / Receive
[TCG SET Specify Allowed Key Encryption Keys UIDs
for the KPIO Managed Namespace ]

Status

NVMe Security Send / Receive
[TCG GET KPIO Public Key Certificate ]

Status

GET Keys’ UIDs from KeyUIDs Key Store

GET [KEK Key UID,
Key Wrapping Specification
(Encrypt with KPIO Public Key)]

GET [MEK Key UID,
Key Wrapping Specification
(Encrypt with KEK(Key UID))]

NVMe Security Send / Receive
[TCG KMIP IMPORT Keys ]

- Batch all KEKs & MEKs in a single KMIP message to the drive

Authenticate & Unwrap, Extract Keys and their attributes, Load Key Cache

Status

**NOTE:**
Exact command tokenization details can be found in the TCG Key Per I/O Application Note

**KPIO’s KMIP BATCH ITEM FORMAT**

```
<RequestHeader>
    ...
    <BatchCount ....value="N Number of Keys"/>
</RequestHeader>
<BatchItem> <!-- Per Key -->
    <OperationType....value="IMPORT"/>
    <RequestPayload>
        <ObjectType...value="Symmetric Key">
            <Attribute>
                <Attribute #1/>
                ...
                <Attribute #N/>
            </Attribute>
        <SymmetricKey>
            ....
            <Keyvalue.....value="XX....XX"/>
            <KeyWrappingData>
                <WrappingMethod Type....value="Encrypt"/>
                <EncryptionKeyInformation>
                    <UniqueIdentifier......value = "Wrapping Key UID"/>
                    <CryptographicParameters>
                        <Param #1/>
                            ....
                        <Param #N/>
                    </CryptographicParameters>
                </EncryptionKeyInformation>
            </KeyWrappingData>
        </SymmetricKey>
    </RequestPayload>
</BatchItem>
```

**KPIO’s Key Type-Related Attributes (e.g., NSID, Key Tag, TCG UIDs, etc..)**

SD’s Volatile Key Cache/Table State On Cmd Completion (Example)

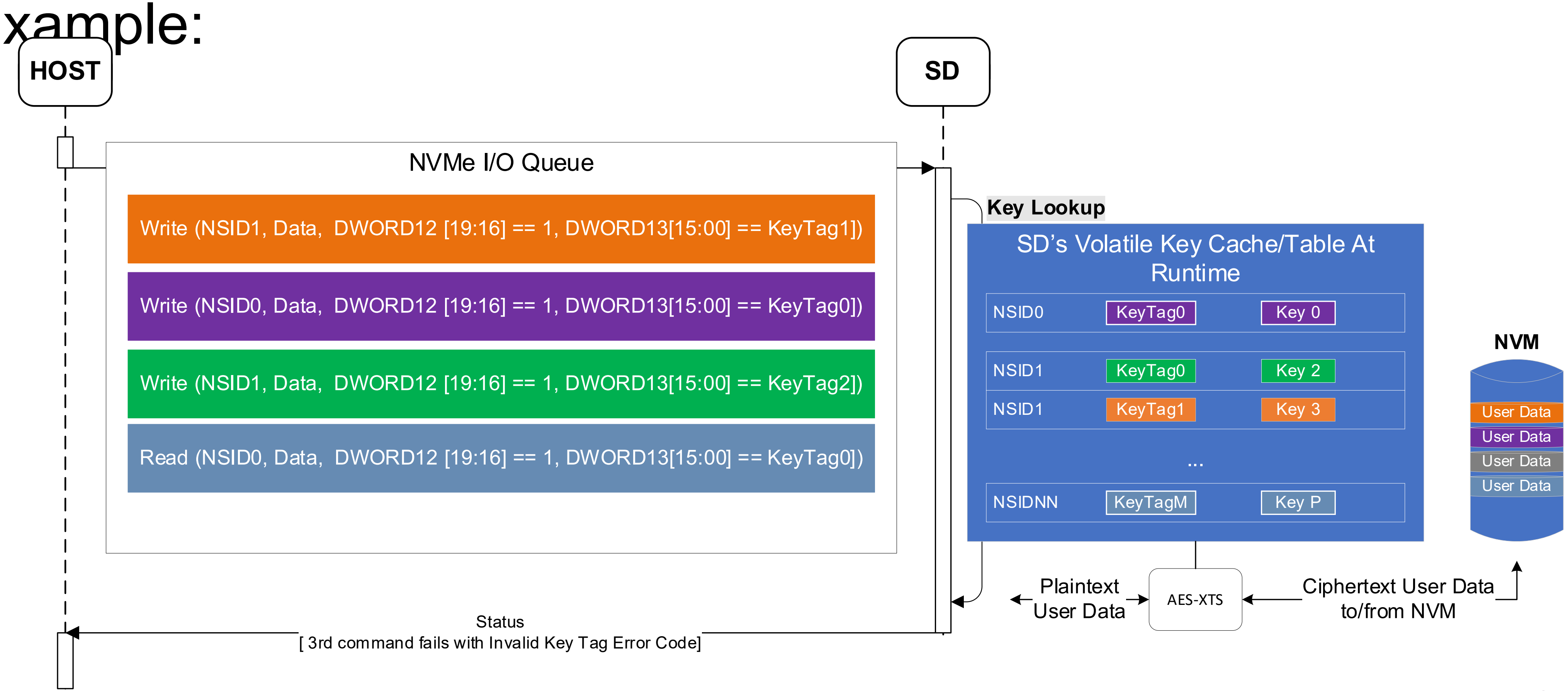
| | | |
|---|---|---|
| NSID0 | KeyTag0 | Key 0 |
| NSID1 | KeyTag0 | Key 2 |
| NSID1 | KeyTag1 | Key 3 |
| | ... | |
| NSIDNN | KeyTagM | Key P |

# Host Management of the SD's Key Cache: Selecting MEKs to Use During I/O

- NVMe® TP4055 defines new KPIO-related Command Extension Type (CETYPE) in DWORD12 and Command Extension Value (CEV) in DWORD13 fields for all read and write I/O commands to indicate to the Storage Device:
  - Key Tag Presence (CETYPE != 0).
  - Key Tag Value (CEV == KEYTAG) associated with MEK to be used for encryption or decryption of data in that I/O command.

# Host Management of the SD’s Key Cache: Selecting MEKs to Use During I/O

- Read/Write IO Example:



NVMe I/O Queue

Write (NSID1, Data,  DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag1])

Write (NSID0, Data,  DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag0])

Write (NSID1, Data,  DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag2])

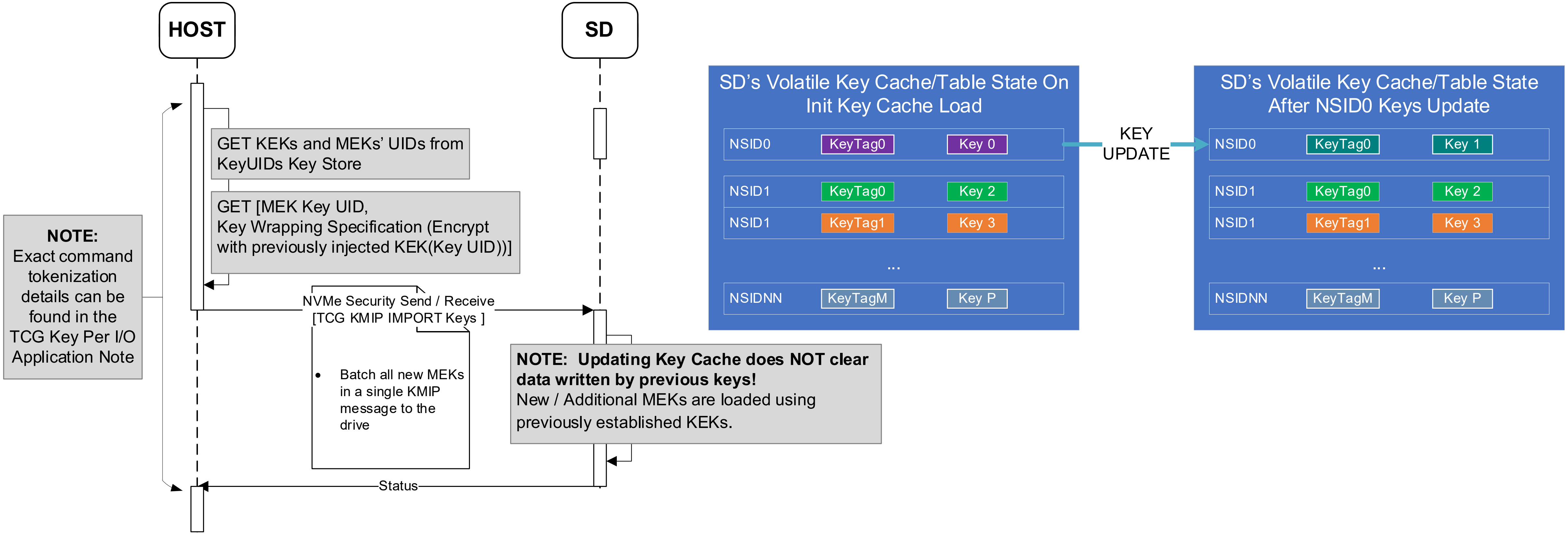
Read (NSID0, Data,  DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag0])

SD’s Volatile Key Cache/Table At Runtime

| | | |
|---|---|---|
| NSID0 | KeyTag0 | Key 0 |
| NSID1 | KeyTag0 | Key 2 |
| NSID1 | KeyTag1 | Key 3 |
| | ... | |
| NSIDNN | KeyTagM | Key P |

Status
[ 3rd command fails with Invalid Key Tag Error Code]

# Host Management of the SD’s Key Cache: Updating the Key Cache


HOST
SD
GET KEKs and MEKs’ UIDs from KeyUIDs Key Store
GET [MEK Key UID, Key Wrapping Specification (Encrypt with previously injected KEK(Key UID))]
NVMe Security Send / Receive [TCG KMIP IMPORT Keys ]
Status
KEY UPDATE

- Batch all new MEKs in a single KMIP message to the drive

**NOTE:** Exact command tokenization details can be found in the TCG Key Per I/O Application Note

**NOTE: Updating Key Cache does NOT clear data written by previous keys!**
New / Additional MEKs are loaded using previously established KEKs.

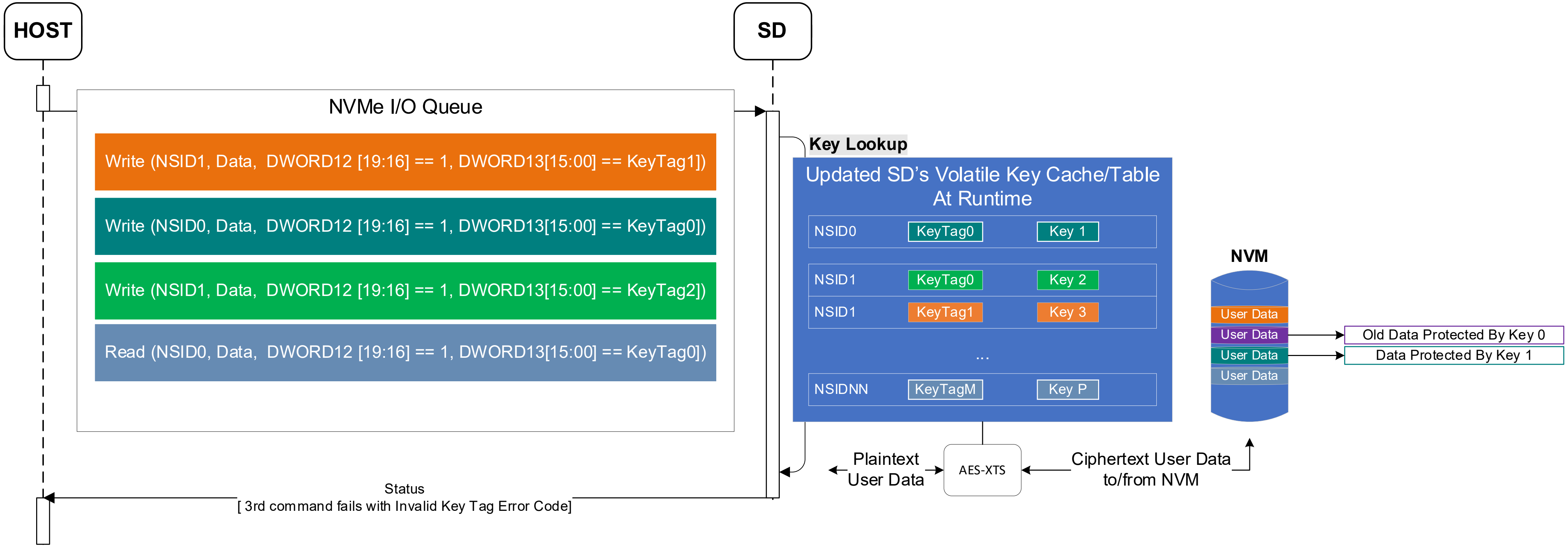
SD’s Volatile Key Cache/Table State On Init Key Cache Load

| | | |
|---|---|---|
| NSID0 | KeyTag0 | Key 0 |
| NSID1 | KeyTag0 | Key 2 |
| NSID1 | KeyTag1 | Key 3 |
| | ... | |
| NSIDNN | KeyTagM | Key P |

SD’s Volatile Key Cache/Table State After NSID0 Keys Update

| | | |
|---|---|---|
| NSID0 | KeyTag0 | Key 1 |
| NSID1 | KeyTag0 | Key 2 |
| NSID1 | KeyTag1 | Key 3 |
| | ... | |
| NSIDNN | KeyTagM | Key P |

# Host Management of the SD’s Key Cache: Selecting new MEKs to Use During I/O



Write (NSID1, Data,  DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag1])

Write (NSID0, Data,  DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag0])

Write (NSID1, Data,  DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag2])

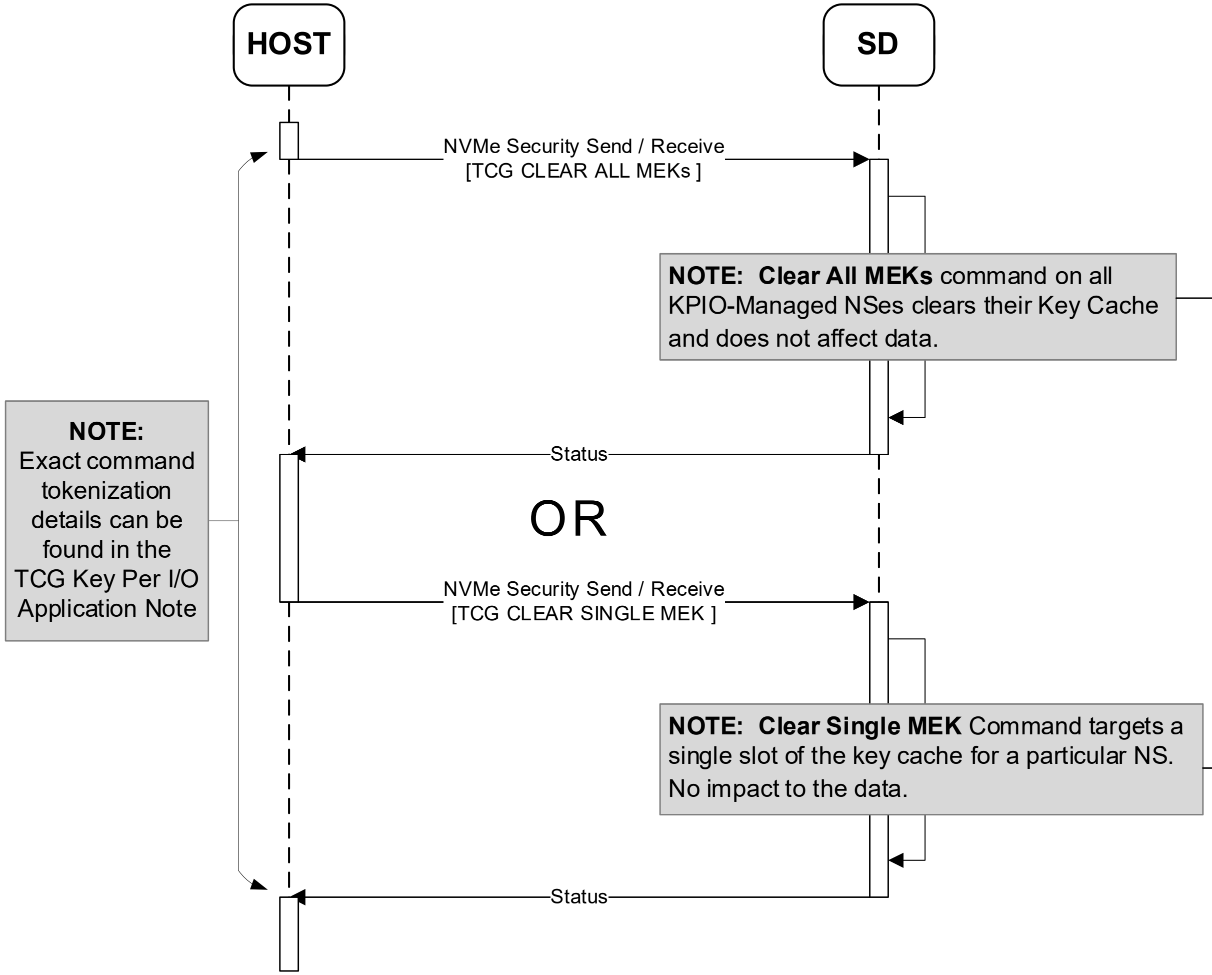
Read (NSID0, Data,  DWORD12 [19:16] == 1, DWORD13[15:00] == KeyTag0])

Updated SD’s Volatile Key Cache/Table At Runtime

| | | |
|---|---|---|
| NSID0 | KeyTag0 | Key 1 |
| NSID1 | KeyTag0 | Key 2 |
| NSID1 | KeyTag1 | Key 3 |
| | ... | |
| NSIDNN | KeyTagM | Key P |

Status
[ 3rd command fails with Invalid Key Tag Error Code]

# Host Management of the SD’s Key Cache: Locking the Key Cache (All NSes vs. Per NS Locking)



**NOTE:** Exact command tokenization details can be found in the TCG Key Per I/O Application Note

**NOTE: Clear All MEKs** command on all KPIO-Managed NSes clears their Key Cache and does not affect data.

**NOTE: Clear Single MEK** Command targets a single slot of the key cache for a particular NS. No impact to the data.

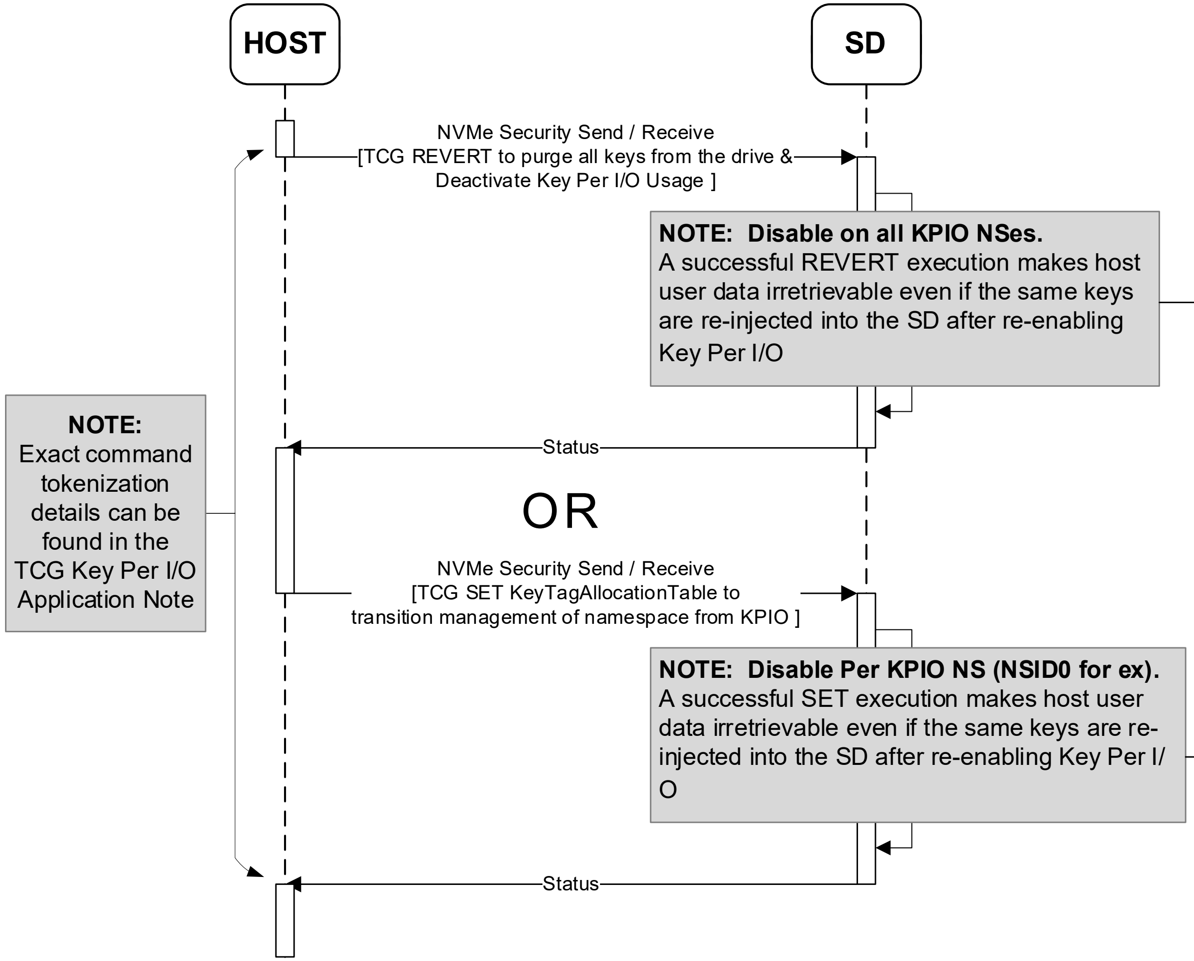
### SD’s Volatile Key Cache/Table State After Clearing all KPIO Keys (Example)

| | | |
|---|---|---|
| NSID0 | KeyTag0 | EMPTY |
| NSID1 | KeyTag0 | EMPTY |
| NSID1 | KeyTag1 | EMPTY |
| | … | |
| NSIDNN | KeyTagM | EMPTY |

### SD’s Volatile Key Cache/Table State After Clearing Single Key (Example)

| | | |
|---|---|---|
| NSID0 | KeyTag0 | Key 1 |
| NSID1 | KeyTag0 | EMPTY |
| NSID1 | KeyTag1 | Key 3 |
| | … | |
| NSIDNN | KeyTagM | Key P |

# Disabling KPIO


HOST
SD
NVMe Security Send / Receive
[TCG REVERT to purge all keys from the drive &
Deactivate Key Per I/O Usage ]
Status
OR
NVMe Security Send / Receive
[TCG SET KeyTagAllocationTable to
transition management of namespace from KPIO ]
Status

**NOTE:** Exact command tokenization details can be found in the TCG Key Per I/O Application Note

**NOTE: Disable on all KPIO NSes.** A successful REVERT execution makes host user data irretrievable even if the same keys are re-injected into the SD after re-enabling Key Per I/O

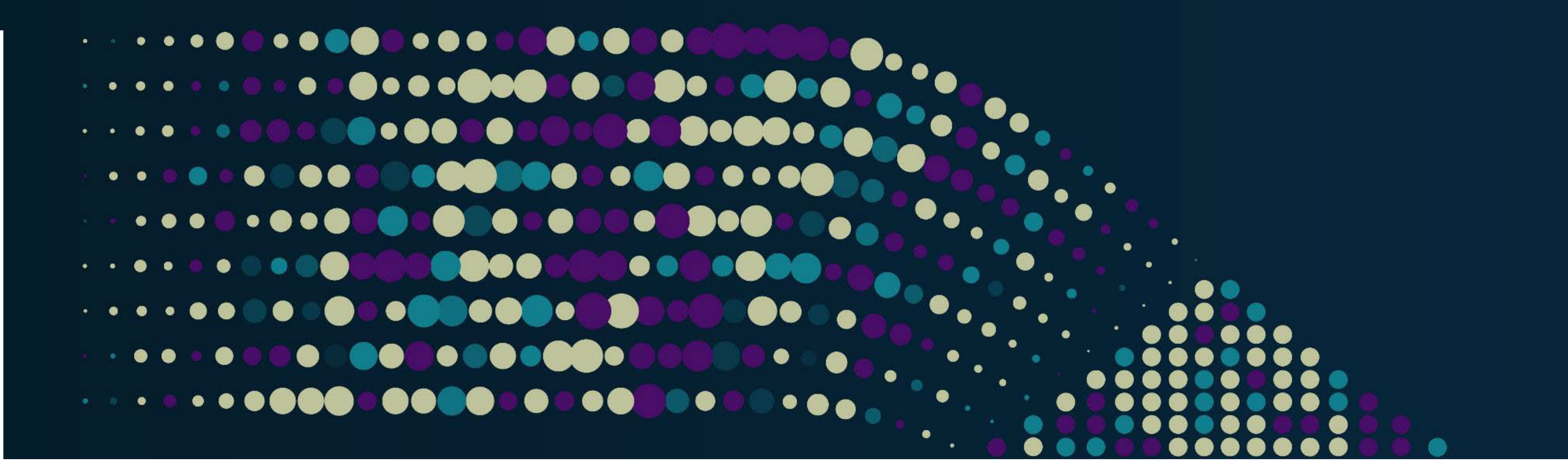
**NOTE: Disable Per KPIO NS (NSID0 for ex).** A successful SET execution makes host user data irretrievable even if the same keys are re-injected into the SD after re-enabling Key Per I/O

SD's Volatile Key Cache/Table State On Cmd Completion (Example)

| | | |
|---|---|---|
| NSID0 | N/A | SD's Generated Key0 |
| NSID1 | N/A | SD's Generated Key1 |
| | ... | |
| NSIDNN | N/A | SD's Generated KeyNN |

SD's Volatile Key Cache/Table State On Cmd Completion (Example)

| | | |
|---|---|---|
| NSID0 | N/A | SD's Generated Key0 |
| NSID1 | KeyTag0 | Key 2 |
| NSID1 | KeyTag1 | Key 3 |
| | ... | |
| NSIDNN | KeyTagM | Key P |

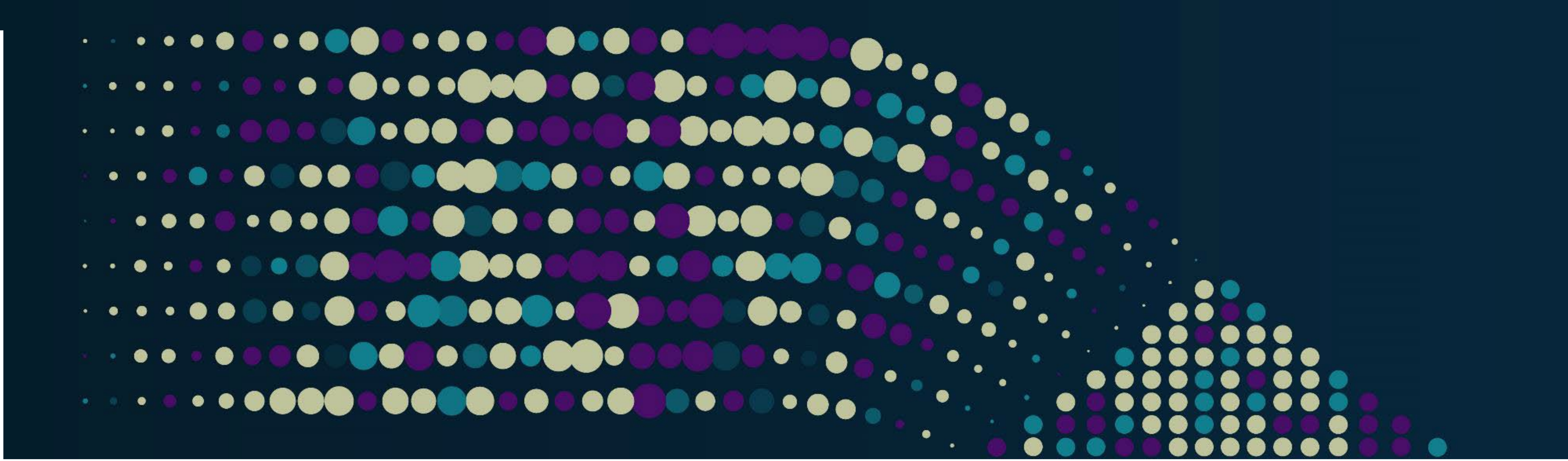


# Summary

Section Subtitle

# Conclusions

- Key Per I/O enabled drives offer another encryption option at the drive level
- External key management allows storage drives to support multiple tenants (VM and containers); may offer customer options for cloud implementations
- Drives impose no limits on the number of MEKs used to protect data; hosts can use large numbers of MEKs (e.g., a unique MEK for each user, file, etc.)



# Please take a moment to rate this session.

Your feedback is important to us.