Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the newly registered object.
Template-Attribute, see 2.1.8	No	An OPTIONAL list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

Table 128: Register Response Payload

If a Managed Cryptographic Object is registered, then the following attributes SHALL be included in the Register request, either explicitly, or via specification of a template that contains the attribute.

Attribute	REQUIRED
Cryptographic Algorithm, see 3.4	Yes, MAY be omitted only if this information is encapsulated in the Key Block. Does not apply to Secret Data. If present, then Cryptographic Length below SHALL also be present.
Cryptographic Length, see 3.5	Yes, MAY be omitted only if this information is encapsulated in the Key Block. Does not apply to Secret Data. If present, then Cryptographic Algorithm above SHALL also be present.
Certificate Length, see 3.9	Yes. Only applies to Certificates.
Cryptographic Usage Mask, see 3.19	Yes.
Digital Signature Algorithm, see 3.16	Yes, MAY be omitted only if this information is encapsulated in the Certificate object. Only applies to Certificates.

Table 129: Register Attribute Requirements

4.4 Re-key

This request is used to generate a replacement key (with new key material) for an existing symmetric key and to adjust key lifecycle attributes in the replacement key. Although the API permits the caller to supply attributes on this request either directly or through Template references (see 4 for how the server SHALL process such requests), if the caller supplies an Offset, then only the UsageLimit is relevant and the server SHOULD return an error (see 11.5) if any other attributes are passed. However, if the caller does not supply an Offset, then only those attributes that pertain to active key lifecycle (i.e., Activation Date, Process Start Date, Protect Stop Date, Deactivation Date, and UsageLimit) are relevant, and the server SHOULD ignore all others passed by the caller. Attributes on the existing symmetric key other than those previously mentioned SHALL be copied to the replacement key~~It is analogous to the Create operation,~~

~~except that attributes of the replacement key are copied from the existing key~~, with the exception of the attributes discussed below.~~listed in Table 131.~~

As the replacement key takes over the name attribute of the existing key, Re-key SHOULD only be performed once on a given key.

The server SHALL copy the Unique Identifier of the replacement key returned by this operation into the ID Placeholder variable.

As a result of Re-key, the Link attribute of the existing key is set to point to the replacement key and vice versa.

An *Offset* MAY be used to indicate the difference between the Initialization Date and the Activation Date of the replacement key. If no Offset is specified, the Activation Date, Process Start Date, Protect Stop Date and Deactivation Date values SHALL be~~are~~ copied from the existing key unless the caller passes in replacement values for these attributes. If an Offset is specified~~set~~ and dates exist for the existing key, then the dates of the replacement key SHALL be set based on the dates of the existing key as follows:

Attribute in Existing Key	Attribute in Replacement Key
Initial Date (IT_1)	Initial Date (IT_2) > IT_1
Activation Date (AT_1)	Activation Date (AT_2) = IT_2+ *Offset*
Process Start Date (CT_1)	Process Start Date = CT_1+(AT_2- AT_1)
Protect Stop Date (TT_1)	Protect Stop Date = TT_1+(AT_2- AT_1)
Deactivation Date (DT_1)	Deactivation Date = DT_1+(AT_2- AT_1)

Table 130: Computing New Dates from Offset during Re-key

Attributes that are not copied from the existing key and are handled in a specific way for the replacement key are:

Attribute	Action
Initial Date, see 3.23	Set to the current time
Destroy Date, see 3.28	Not set
Compromise Occurrence Date, see 3.29	Not set
Compromise Date, see 3.30	Not set
Revocation Reason, see 3.31	Not set
Unique Identifier, see 3.1	New value generated
Usage Limits, see 3.21	If caller supplies a desired value, then it is applied to the new key. Otherwise, t~~T~~he Total value is copied from the existing key. In either case, ~~and~~ the Count value is set to the Total value in the existing key.
Name, see 3.2	Set to the name(s) of the existing key; all name attributes are removed from the existing key.
State, see 3.22	Set based on attributes values, such as dates, as shown in Table 130~~Table 130~~
Digest, see 3.16	Recomputed from the replacement key value
Link, see 3.35	Set to point to the existing key as the replaced key
Last Change Date, see 3.38	Set to current time
Fresh, see 3.34	Set to TRUE in replacement object

1170 *Table 131: Re-key Attribute Requirements*

Formatted: Font: 10 pt

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	Determines the existing Symmetric Key being re-keyed. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Offset	No	An Interval object indicating the difference between the Initialization Date and the Activation Date of the replacement key to be created.
Template-Attribute, see 2.1.8	No	Specifies desired object attributes using templates and/or individual attributes.

Table 132: Re-key Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the newly-created replacement Symmetric Key.
Template-Attribute, see 2.1.8	No	An OPTIONAL list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

Table 133: Re-key Response Payload

4.5 Re-key Key Pair

This request is used to generate a replacement key pair (with new key material for each key in the pair) for an existing public/private key pair, and to simultaneously adjust key lifecycle attributes in the replacement keys. Although the API permits the caller to supply attributes on this request either directly or through Template references (see 4 for how the server SHALL process such requests), if the caller supplies an Offset, then only the UsageLimit(s) is(are) relevant and the server SHOULD return an error (see 11.5) if any other attributes are passed. However, if the caller does not supply an Offset, then only those attributes that pertain to active key lifecycle (i.e., Activation Date, Process Start Date, Protect Stop Date, Deactivation Date, and UsageLimit) are relevant, and the server SHOULD ignore all others passed by the caller. Attributes on the existing keys other than those previously mentioned SHALL be copied to the replacement key~~It is analogous to the Create Key Pair operation, except that attributes of the replacement key pair are copied from the existing key pair~~, with the exception of the attributes discussed below.~~listed in Table 135.~~

As the replacement of the key pair takes over the name attribute for the existing public/private key pair, Re-key Key Pair SHOULD only be performed once on a given key pair.

As a result of the Re-key Key Pair operation the Link Attribute for both the existing public key and private key objects are updated to point to the replacement public and private key, respectively, and vice-versa .

The server SHALL copy the Private Key Unique Identifier of the replacement private key returned by this operation into the ID Placeholder variable.

An Offset MAY be used to indicate the difference between the Initialization Date and Activation Date of the replacement key pair. If no Offset is specified, the Activation Date, Process Start Date, Protect Stop Date and Deactivation Date values SHALL be copied from the existing key pair unless the caller passes in replacement values for these attributes. If ~~the~~ an Offset is ~~set~~ specified and the dates exist for the existing key pair, then the dates of the replacement key pair SHALL be set based on the dates of the existing key pair as follows:

Attribute in Existing Key Pair	Attribute in Replacement Key Pair
Initial Date (IT_1)	Initial Date (IT_2) > IT_1
Activation Date (AT_1)	Activation Date (AT_2) = IT_2+ *Offset*
Deactivation Date (DT_1)	Deactivation Date = DT_1+(AT_2- AT_1)

Table 134: Computing New Dates from Offset during Re-key Key Pair

Attributes that are not copied from the existing key pair and which are handled in a specific way are:

Attribute	Action
Private Key Unique Identifier, see 3.1	New value generated
Public Key Unique Identifier, see 3.1	New value generated
Name, see 3.2	Set to the name(s) of the existing public/private keys; all name attributes of the existing public/private keys are removed.
Digest, see 3.17	Recomputed for both replacement public and private keys from the new public and private key values
Usage Limits, see 3.21	If caller supplies desired values, then they are applied to the new keys. Otherwise, t~~T~~he Total Bytes/Total Objects value is copied from the existing key pair. In either case,~~,~~ ~~while~~ the Byte Count/Object Count values are set to the Total Bytes/Total Objects in the existing key pair..
State, see 3.22	Set based on attributes values, such as dates, as shown in Table 134~~Table xx~~
Initial Date, see 3.23	Set to the current time
Destroy Date, see 3.28	Not set
Compromise Occurrence Date, see 3.29	Not set
Compromise Date, see 3.30	Not set
Revocation Reason, see 3.31	Not set
Link, see 3.35	Set to point to the existing public/private keys as the replaced public/private keys
Last Change Date, see 3.38	Set to current time
Fresh, see 3.34	Set to TRUE in replacement objects

Table 135: Re-key Key Pair Attribute Requirements

Request Payload		
Object	**REQUIRED**	**Description**
Private Key Unique Identifier, see 3.1	No	Determines the existing Asymmetric key pair to be re-keyed. If omitted, then the ID Placeholder is substituted by the server.
Offset	No	An Interval object indicating the difference between the Initialization date and the Activation Date of the replacement key pair to be created.
Common Template-Attribute, see 2.1.8	No	Specifies desired attributes in templates and/or as individual attributes that apply to both the Private and Public Key Objects.
Private Key Template-Attribute, see 2.1.8	No	Specifies templates and/or attributes that apply to the Private Key Object. Order of precedence applies.
Public Key Template-Attribute, see 2.1.8	No	Specifies templates and/or attributes that apply to the Public Key Object. Order of precedence applies.

Table 136: Re-key Key Pair Request Payload

For multi-instance attributes, the union of the values found in the templates and attributes of the Common, Private, and Public Key Template-Attribute is used. For single-instance attributes, the order of precedence is as follows:

1. attributes specified explicitly in the Private and Public Key Template-Attribute, then
2. attributes specified via templates in the Private and Public Key Template-Attribute, then
3. attributes specified explicitly in the Common Template-Attribute, then
4. attributes specified via templates in the Common Template-Attribute

If there are multiple templates in the Common, Private, or Public Key Template-Attribute, then the subsequent value of the single-instance attribute takes precedence.

Response Payload		
Object	**REQUIRED**	**Description**
Private Key Unique Identifier, see 3.1	Yes	The Unique Identifier of the newly created replacement Private Key object.
Public Key Unique Identifier, see 3.1	Yes	The Unique Identifier of the newly created replacement Public Key object.
Private Key Template-Attribute, see 2.1.8	No	An OPTIONAL list of attributes, for the Private Key Object, with values that were not specified in the request, but have been implicitly set by the key management server.

Public Key Template-Attribute, see 2.1.8	No	An OPTIONAL list of attributes, for the Public Key Object, with values that were not specified in the request, but have been implicitly set by the key management server.

Table 137: Re-key Key Pair Response Payload

4.6 Derive Key

This request is used to derive a symmetric key or Secret Data object from a key or secret data that is already known to the key management system. The request SHALL only apply to Managed Cryptographic Objects that have the Derive Key bit set in the Cryptographic Usage Mask attribute of the specified Managed Object (i.e., are able to be used for key derivation). If the operation is issued for an object that does not have this bit set, then the server SHALL return an error. For all derivation methods, the client SHALL specify the desired length of the derived key or Secret Data object using the Cryptographic Length attribute. If a key is created, then the client SHALL specify both its Cryptographic Length and Cryptographic Algorithm. If the specified length exceeds the output of the derivation method, then the server SHALL return an error. Clients MAY derive multiple keys and IVs by requesting the creation of a Secret Data object and specifying a Cryptographic Length that is the total length of the derived object. The length SHALL NOT exceed the length of the output returned by the chosen derivation method.

The fields in the request specify the Unique Identifiers of the keys or Secret Data objects to be used for derivation (e.g., some derivation methods MAY require multiple keys or Secret Data objects to derive the result), the method to be used to perform the derivation, and any parameters needed by the specified method. The method is specified as an enumerated value. Currently defined derivation methods include:

- *PBKDF2* – This method is used to derive a symmetric key from a password or pass phrase. The PBKDF2 method is published in **[PKCS#5]** and **[RFC2898]**.
- *HASH* – This method derives a key by computing a hash over the derivation key or the derivation data.
- *HMAC* – This method derives a key by computing an HMAC over the derivation data.
- *ENCRYPT* – This method derives a key by encrypting the derivation data.
- *NIST800-108-C* – This method derives a key by computing the KDF in Counter Mode as specified in **[SP800-108]**.
- *NIST800-108-F* – This method derives a key by computing the KDF in Feedback Mode as specified in **[SP800-108]**.
- *NIST800-108-DPI* – This method derives a key by computing the KDF in Double-Pipeline Iteration Mode as specified in **[SP800-108]**.
- *Extensions*

The server SHALL perform the derivation function, and then register the derived object as a new Managed Object, returning the new Unique Identifier for the new object in the response. The server SHALL copy the Unique Identifier returned by this operation into the ID Placeholder variable.

As a result of Derive Key, the Link attributes (i.e., Derived Key Link in the objects from which the key is derived, and the Derivation Base Object Link in the derived key) of all objects involved SHALL be set to point to the corresponding objects.