# Mechanism Definition

CKM\_SEAL\_KEY

The mechanism is used to wrap and unwrap key material in an implementation specific manner. When used with a special class of wrap/unwrap keys, it is even possible to wrap keys that would normally be unextractable when the wrapping key is one of the defined seal keys (see [section on global objects, seal keys]).

The actual cryptographic mechanism used to perform the wrap and unwrap operations is implementation dependent, and, while the wrapping key must be a CKO\_SECRET\_KEY, it may be any type of secret key. Whatever mechanism is selected by the vendor MUST integrity protect the wrapped key, and MUST refuse to unwrap any key where the integrity fails to verify.

There are no parameters for this mechanism.

This mechanism provides a standard method for moving sensitive key material to storage external to the token, but where the material is still cryptographically tied to the token.

When used with one of the seal keys, the mechanism will allow C\_WrapKey to wrap any existing key, whether exportable (CKA\_EXTRACTABLE = TRUE) or not, and will output an opaque byte array. The information wrapped includes all pertinent key attributes in an implementation dependent form. "Pertinent" here means all attributes that are actually meaningful for the key - e.g. don't include CKA\_MODULUS for an CKK\_EC key that are present on the key with a non-default value.

The unwrap operation reverses the wrap operation and MUST produce exactly the same object as was wrapped with the single exception of key handle value. This mechanism DOES NOT support unwrapping keys on any token besides the token from which the key was originally wrapped.

The mechanism SHOULD refuse to wrap a key when the strength of the wrapping key is less than that of the key to be wrapped.

When used with any key other than one of the seal keys, the normal restrictions on wrapping MUST be enforced.

When this mechanism is used with C\_UnwrapKey, ulAttributeCount should be 0. Any specified attributes are silently ignored.

When CKM\_SEAL\_KEY is used as the mechanism for C\_UnwrapKey, and the unwrapping key is one of the token seal keys, the CKA\_EXTRACTABLE, CKA\_NEVER\_EXTRACTABLE, CKA\_LOCAL, CKA\_SENSITIVE, and CKA\_ALWAYS\_SENSITIVE attributes are set from the data included with the wrapped key. If used with any other key, then those attributes are set as described in the C\_UnwrapKey documentation.

# Example of Use

CK\_SESSION\_HANDLE hSession;

CK\_OBJECT\_HANDLE hSealKey;

CK\_OBJECT\_HANDLE hKeyToBeWrapped;

CK\_ULONG ulObjectCount;

CK\_RV rv;

CK\_ATTRIBUTE template[] = {

 {CKA\_GLOBAL, CKV\_TOKEN\_GLOBAL, 4},

 {CKA\_OBJECT\_ID, tokenSealOid, tokenSealOidLen},

 {CKA\_CLASS, CKO\_SECRET\_KEY, 4}};

CK\_MECHANISM sealMech = { CKM\_SEAL\_KEY, NULL\_PTR, 0 };

CK\_BYTE\_PTR pWrappedKey;

CK\_ULONG ulWrappedKeyLength = 0;

.

.

// Find the seal key I want to use

rv = C\_FindObjectsInit (hSession, template, 3);

// check return

rv = C\_FindObjects (hSession, &hSealKey, 1, &ulObjectCount);

//check return

rv = C\_FindObjectsFinal(hSession);

// check return

// get buffer length for wrapped key

rv = C\_WrapKey (hSession, &sealMech,

 hSealKey, hKeyToBeWrapped,

 NULL\_PTR, &ulWrappedKeyLength);

pWrappedKey = malloc (ulWrappedKeyLength);

rv = C\_WrapKey (hSession, &sealMech,

 hSealKey, hKeyToBeWrapped,

 pWrappedKey, &ulWrappedKeyLength);