# Global Attribute Values

CKA\_GLOBAL - CK\_GLOBAL\_TYPES

This attribute is used to identify objects that generally persist through reinitializations of individual tokens, but are created though token or driver action and not user action. The value of this attribute for all user created objects, and the default value of the attribute, is CKV\_NOT\_GLOBAL.

This attribute may not be used in any template for any function used to create an object, and may not be specified in a C\_SetAttributeValue template. Objects where CKA\_GLOBAL is anything but CKV\_NOT\_GLOBAL are created and managed by the token implementation, or by the PKCS11 driver or hardware implementation. Users cannot create global objects directly.

The collection of values for CKA\_GLOBAL supports a model where a single PKCS11 driver instantiation may provide access to multiple PKCS11 tokens.

typedef CK\_GLOBAL\_TYPES CK\_ULONG;

#define CKV\_NOT\_GLOBAL 0

All user created objects have this value.

#define CKV\_TOKEN\_GLOBAL 1

Global objects which are specific to a token have this value. .

Some examples of CKV\_TOKEN\_GLOBAL objects might be a token identity key pair associated with a token identity certificate.

#define CKV\_COLLECTION\_GLOBAL 2

Global objects which are unique to a specific collection of tokens have this value. This may represent a logical grouping of tokens, or a single physical device which presents as a group of tokens.

An example of a CKV\_COLLECTION\_GLOBAL object might be a platform credential such as a certificate binding information about the hardware and software implementation of a device which acts as a collection of PKCS11 tokens.

#define CKV\_IMPLEMENTATION\_GLOBAL 3

Global objects which are specific to a PKCS11 driver implementation have this value.

An example of a CKV\_IMPLEMENTATION\_GLOBAL object might be a driver credential (e.g. certificate) binding the hash of the driver executable to the platform or manufacturer.

#define CKV\_CLIENT\_GLOBAL 4

Global objects which are implemented on the client side outside of the device specific PKCS11 driver as a PKCS11 pseudo-object have this value.

This type of global object may be applicable for an implementation of a PKCS11 composite driver which allows access to tokens and collections of tokens of different manufacturers through a single driver.

All objects where CKA\_GLOBAL != CKV\_NOT\_GLOBAL MUST also have an attribute CKA\_OBJECT\_ID to define the specific use of the global object.

The combination of CKA\_OBJECT\_ID and CKA\_CLASS MUST be unique for any given token global object (CKA\_GLOBAL = CKV\_TOKEN\_GLOBAL) and identifies the same object regardless of which session accesses the object on that token.

The combination of CKA\_OBJECT\_ID and CKA\_CLASS MUST be unique for any given collection global object (CKA\_GLOBAL == CKV\_COLLECTION\_GLOBAL) and identifies the same object regardless of which session or token accesses the object for a given collection of tokens. This combination MAY represent different objects for any two tokens which are not members of the same collection.

The combination of CKA\_OBJECT\_ID and CKA\_CLASS MUST be unique for any given implementation global object and identifies the same object regardless of which session or token accesses the object.

The uniqueness rules above should be read as permitting a collection of only a single public key, single private key, single data object, and/or single certificate per a CKA\_OBJECT\_ID value for any given grouping.

# Global Objects

## Identity Objects

### Token Identity

* CKA\_OBJECT\_ID = <token id oid>
* CKA\_GLOBAL = CKV\_TOKEN\_GLOBAL
* CKA\_CLASS = CKO\_PRIVATE\_KEY, CKO\_PUBLIC\_KEY, CKO\_CERTIFICATE
* CKA\_PRIVATE= FALSE

The private key has the additional following attributes:

* CKA\_EXTRACTABLE = FALSE
* CKA\_COPYABLE = FALSE
* CKA\_DECRYPT = FALSE
* CKA\_SIGN = TRUE
* CKA\_VERIFY/VERIFY\_RECOVER/SIGN\_RECOVER/ENCRYPT/DECRYPT/WRAP/UNWRAP/DERIVE= FALSE
* CKA\_SENSITIVE = TRUE

The certificate has the additional following attributes:

* CKA\_CERTIFICATE\_TYPE = CKC\_X\_509

This is a group of up to three distinct objects: a private key, an optional public key that matches the private key, and an optional X.509 certificate binding that public key. The public key may be of any type supported by the platform. The certificate uniquely identifies a specific token. The certificate SHALL be signed by the manufacturer of the token.

When it defines a private key, it identifies a private key that is unique to this token, is non extractable, and is usable by any user of the token including the SO (CKA\_PRIVATE = FALSE). The private key MUST be restricted to the use of signature mechanisms and those mechanisms should include CKM\_CERTIFY\_KEY.

TBD - contents and structure of the certificate. May use the SubjectAltName construct of manufacturer/serial number[/token number]

### Collection Identity

This set of objects is identical to the those described above under Token Identity with two exceptions:

* CKA\_OBJECT\_ID = <collection id oid>
* CKA\_GLOBAL = CKV\_COLLECTION\_GLOBAL

As a collection global object, this represents a collection of tokens rather than an individual token. This can be used where a single device (e.g. a hardware security module) provides multiple virtual tokens.

## Platform Information

### Certificate

* CKA\_OBJECT\_ID = <platform id oid>
* CKA\_GLOBAL = CKV\_COLLECTION\_GLOBAL
* CKA\_CERTIFICATE\_TYPE = CKC\_X\_509

This certificate is signed by the manufacturer and identifies the class of hardware, software and firmware that make up the underlying security module.

TBD - contents and structure of the certificate.

### Measurement

TBD - This could be a signed object representing a measurement (in Trusted Platform Module terms) of the underlying security module.

## Seal Objects

Seal objects are always secret keys. They are used to "export" or wrap keys that normally couldn't be exported, mainly to free up internal space. They are never exportable, they are always private (e.g. usable only when a token user is logged in). They may only be used with the CKM\_SEAL\_KEY mechanism.

Cryptographically, these keys, when used with the CKM\_SEAL\_KEY mechanism allow the use of external storage, while still protecting the keys as if they were internal. The sealed keys may only be unwrapped by the same token (or same session) that wrapped them.

Since the seal keys are zeroized upon session close or token zeroization, any sealed key material external to the token becomes no better than random bits upon those actions.

As general guidance, the seal key should be greater than or equal to in strength to any key it wraps or unwraps.

### Token Seal Key

* CKA\_GLOBAL = CKV\_TOKEN\_GLOBAL
* CKA\_OBJECT\_ID = <token seal key oid>
* CKA\_CLASS = CKO\_SECRET\_KEY
* CKA\_TOKEN = TRUE
* CKA\_PRIVATE=TRUE
* CKA\_SENSITIVE=TRUE

This key is persistent across sessions, is generated when the token is initialized, but is zeroized and replaced if the token is re-initialized.

### Session seal key

* CKA\_GLOBAL = CKV\_TOKEN\_GLOBAL
* CKA\_OBJECT\_ID = <session seal key oid>
* CKA\_CLASS = CKO\_SECRET\_KEY
* CKA\_TOKEN = FALSE
* CKA\_PRIVATE=TRUE
* CKA\_SENSITIVE=TRUE

This key is valid only during a specific session and is generated at the beginning of each session and zeroized at the conclusion of that session.