eSMART®

Список функций PKCS11

1. Таблица функций

Некоторые особенности реализации:

- *библиотека поддерживает объекты типа* `CKO_DATA, CKO_CERTIFICATE, CKO_PUBLIC_KEY, CKO_PRIVATE_KEY, CKO_SECRET_KEY;`
- *объекты* `CKO_CERTIFICATE и CKO_PUBLIC_KEY` *защищены PIN-кодом пользователя только на запись. Соответственно, чтение значения возможно без предъявления PIN-кода, в том числе, для PRIVATE-объектов;*

GENERAL-PURPOSE FUNCTIONS		
`C_Initialize`	*Инициализация библиотеки*	
`C_Finalize`	*Вызывается при завершении работы с библиотекой*	
`C_GetInfo`	*Получение информации о библиотеке, например, версия и наименование разработчика*	
`C_GetFunctionList`	*Получение адресов функций*	
SLOT AND TOKEN MANAGEMENT FUNCTIONS		
`C_GetSlotList`	*Получение списка слотов*	
`C_GetSlotInfo`	*Получение информации о слоте*	
`C_GetTokenInfo`	*Получение информации о токене*	
`C_WaitForSlotEvent`[1]	*Ожидание события «прихода/ухода» токена*	
`C_GetMechanismList`	*Получение списка поддерживаемых механизмов*	
`C_GetMechanismInfo`	*Получение параметров механизма*	
`C_InitToken`	*Инициализация токена*	*В процессе инициализации устанавливается PIN-код пользователя, что не соответствует спецификации и сделано для совместимости с некоторыми существующими решениями.*
`C_InitPIN`	*Инициализация PIN-кода пользователя*	*Так как PIN-код пользователя устанавливается при выполнении* `C_InitToken` *эту функцию вызывать не нужно. В случае вызова будет возвращена ошибка.*
`C_SetPIN`	*Установка нового PIN-кода администратора или пользователя*	
`C_OpenSession`	*Открытие сессии*	
`C_CloseSession`	*Закрытие сессии*	
`C_CloseAllSessions`	*Закрытие всех сессий*	
`C_GetSessionInfo`	*Получение информации о сессии*	

[1] C_WaitForSlotEvent *работает только с установленным флагом* CKF_DONT_BLOCK *параметра* flags

C_GetOperationState[2]	*Получение текущего состояния крипто-операций*	
C_SetOperationState	*Установка состояния крипто-операций*	
C_Login	*Аутентификация*	
C_Logout	*Деаутентификация*	
OBJECT MANAGEMENT FUNCTIONS		
C_CreateObject	*Создание объектов*	
C_CopyObject	*Копирование объектов*	*Реализовано копирование без изменения атрибутов*
C_DestroyObject	*Удаление объектов*	
C_GetObjectSize	*Получение примерного размера памяти, занимаемой объектом*	
C_GetAttributeValue	*Чтение значения атрибута*	
C_SetAttributeValue	*Установка значения атрибута*	
C_FindObjectsInit	*Функции поиска объектов*	
C_FindObjects		
C_FindObjectsFinal		
ENCRYPTION FUNCTIONS		
C_EncryptInit	*Функции шифрования*	*Поддерживаемые механизмы:* *CKM_RSA_PKCS* *CKM_RSA_X_509* *CKM_AES_ECB* *CKM_AES_CBC* *CKM_AES_CBC_PAD* *CKM_DES_ECB* *CKM_DES_CBC* *CKM_DES_CBC_PAD* *CKM_DES3_ECB* *CKM_DES3_CBC* *CKM_DES3_CBC_PAD* *CKM_GOST28147*[3] *CKM_GOST28147_ECB*
C_Encrypt		
C_EncryptUpdate		
C_EncryptFinal		
DECRYPTION FUNCTIONS		
C_DecryptInit	*Функции дешифрования*	*Поддерживаемые механизмы, как для функции шифрования*
C_Decrypt		
C_DecryptUpdate		
C_DecryptFinal		

2 *Функции не поддерживаются*

3 *Механизмы поддерживаются только ESMART Token ГОСТ*

MESSAGE DIGESTING FUNCTIONS		
C_DigestInit	*Функции вычисления хеш-значений*	*Поддерживаемые механизмы:* *CKM_MD5* *CKM_SHA_1* *CKM_SHA256* *CKM_GOSTR3411*
C_Digest		
C_DigestUpdate		
C_DigestFinal		
C_DigestKey	*Вычисление хеш-значения секретного ключа*	
SIGNING AND MACING FUNCTIONS		
C_SignInit	*Функции вычисления цифровой подписи*	*Поддерживаемые механизмы:* *CKM_RSA_PKCS* *CKM_GOSTR3410* *CKM_ECDSA*
C_Sign		
C_SignUpdate		
C_SignFinal		
C_SignRecoverInit		
C_SignRecover		
FUNCTIONS FOR VERIFYING SIGNATURES AND MACS		
C_VerifyInit	*Функции проверки цифровой подписи*	*Поддерживаемые механизмы, как для функции вычисления цифровой подписи*
C_Verify		
C_VerifyUpdate		
C_VerifyFinal		
C_VerifyRecoverInit		
C_VerifyRecover		
DUAL-FUNCTION CRYPTOGRAPHIC FUNCTIONS		
C_DigestEncryptUpdate	*Функции, объединяющие выполнение других крипто-функций*	
C_DecryptDigestUpdate		
C_SignEncryptUpdate		
C_DecryptVerifyUpdate		

KEY MANAGEMENT FUNCTIONS		
C_GenerateKey	*Генерация секретного ключа*	*Поддерживаемые механизмы:* *CKM_AES_KEY_GEN* *CKM_DES_KEY_GEN* *CKM_DES2_KEY_GEN* *CKM_GOST28147_KEY_GEN*
C_GenerateKeyPair	*Генерация ключевой пары*	*Поддерживаемые механизмы:* *CKM_RSA_PKCS_KEY_PAIR_GEN* *CKM_EC_KEY_PAIR_GEN* *CKM_GOSTR3410_KEY_PAIR_GEN*
C_WrapKey	*Шифрование секретного или закрытого ключа* [4]	*Поддерживаемые механизмы, как для функции шифрования*
C_UnwrapKey	*Расшифрование и создание нового секретного или закрытого ключа* [4]	*Поддерживаемые механизмы, как для функции шифрования*
C_DeriveKey	*Создание нового секретного ключа на основе существующего* [4]	*Поддерживаемые механизмы:* *CKM_DES_ECB_ENCRYPT_DATA* *CKM_DES_CBC_ENCRYPT_DATA* *CKM_DES3_ECB_ENCRYPT_DATA* *CKM_DES3_CBC_ENCRYPT_DATA* *CKM_AES_ECB_ENCRYPT_DATA* *CKM_AES_CBC_ENCRYPT_DATA*

[4] *Функции C_WrapKey, UnwrapKey и C_DeriveKey поддерживаются только ESMART Token, но не поддерживаются ESMART Token ГОСТ*

RANDOM NUMBER GENERATION FUNCTIONS		
C_SeedRandom	*Установка seed-значения для генератора случайных чисел*	*Возвращает CKR_RANDOM_SEED_NOT_SUPPORTED*
C_GenerateRandom	*Получение случайного числа*	
PARALLEL FUNCTION MANAGEMENT FUNCTIONS		
C_GetFunctionStatus	*Функции управления параллельным выполнением крипто-операций*	
C_CancelFunction		
PROPRIETARY FUNCTIONS		
C_ISBC_ImportX509Certificate	*Импорт сертификата x509 в формате DER*	
C_ISBC_CreateCSR	*Создание запроса на выпуск сертификата*	
C_ISBC_pkcs7Sign	*Подпись данных в соответствии с PKCS#7*	
C_ISBC_pkcs7Verify	*Проверка подписи PKCS#7*	
C_ISBC_InitToken	*Форматирование с указанием доп. условий*	
C_ISBC_GetCryptoProInfo	*Получить информацию о сохраненных контейнерах КриптПро*	
C_ACS_LoadCert	*Создание ключевой пары и сертификата из файла в формате PKCS#12*	
C_ISBC_CheckSM	*Проверка возможности работать с проверкой Secure Messaging*[5]	*Возвращает* *0 – не поддерживается;* *1 – поддерживается, но не активирован;* *2 – поддерживается и активирован*

*Коды ошибок перечислены в приложении A **Manifest constants** стандарта PKCS#11 (документ pkcs-11v2-20.pdf, стр. 375).*

[5] *http://en.wikipedia.org/wiki/Secure_messaging*

2. Собственные функции

В разделе представлено подробное описание функций раздела PROPRIETARY FUNCTIONS

C_ISBC_ImportX509Certificate

```
CK_RV C_ISBC_ImportX509Certificate
(
      CK_SESSION_HANDLE       hSession,
      CK_UTF8CHAR_PTR         pLabel,
      CK_BYTE_PTR             pID,
      CK_ULONG                ulIDLen,
      CK_BYTE_PTR             pData,
      CK_ULONG                ulDataLen
);
```

Возвращаемое значение:
`CKR_OK` – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs-11v2-20.pdf, стр. 375).

Параметры:

`hSession`	*идентификатор сессии;*
`pLabel`	*`CKA_LABEL` для импортируемого сертификата;*
`pID`	*`CKA_ID` для импортируемого сертификата;*
`ulIDLen`	*размер для `pID`;*
`pData`	*содержимое сертификата в формате DER;*
`ulDataLen`	*размер `pData`*

C_ISBC_CreateCSR

```
CK_RV C_ISBC_CreateCSR
(
            CK_SESSION_HANDLE       hSession,
            CK_OBJECT_HANDLE        publicKey,
            CK_CHAR_PTR             dn,
            CK_ULONG                dnLen,
            CK_BYTE_PTR             csr,
            CK_ULONG_PTR            cstLen,
            CK_OBJECT_HANDLE        privateKey,
            CK_CHAR_PTR             attrs,
            CK_ULONG                attrsLen,
            CK_CHAR_PTR             exts,
            CK_ULONG                extsLen
);
```

Возвращаемое значение:

CKR_OK – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs-11v2-20.pdf, стр. 375).

Параметры:

hSession	*идентификатор сессии;*
publicKey	*идентификатор объекта типа CKO_PUBLIC_KEY, для которого создается запрос;*
dn	*Distinguished Name запроса. В параметре передается массив строк, где в первой строке располагается тип поля в текстовом формате или идентификатор объекта (CN - например). Во второй строке должно находиться значение поля в UTF8. Последующие поля передаются в следующих строках. Количество строк должно быть кратно двум;*
dnLen	*количество строк в массиве, на который ссылается параметр dn;*
csr	*указатель на память куда будет помещен содержимое запрос. Память должна быть выделена заранее. Чтобы узнать необходимый размер памяти необходимо вызвать функцию передав NULL в csr. Необходимый размер будет возвращен в параметре cstLen;*
cstLen	*размер буфера памяти под csr;*
privateKey	*идентификатор объекта типа CKO_PRIVATE_KEY для которого создается запрос;*
attrs	*дополнительные атрибуты для включения в запрос. Формат аналогичен формату параметра dn;*
attrsLen	*количество строк в массиве, на который ссылается параметр attrs;*
exts	*расширения для включения в запрос. В параметре передается массив строк, где в первой строке располагается тип поля в текстовом формате или идентификатор объекта (keyUsage - например). Во второй строке должно находиться значение поля в UTF8. В третьей строке должен быть указано требование в данному расширению, 0 – не критическое, 1 – критическое. Последующие поля передаются в следующих строках. Количество строк должно быть кратно трем;*
extsLen	*количество строк в массиве, на который ссылается параметр exts.*

C_ISBC_pkcs7Sign

```
CK_RV C_ISBC_pkcs7Sign
 (
      CK_SESSION_HANDLE       hSession,
      CK_BYTE_PTR             pData,
      CK_ULONG                ulDataLen,
      CK_OBJECT_HANDLE        hSignCertificate,
      CK_BYTE_PTR             pOut,
      CK_ULONG_PTR            pulOutLen,
      CK_OBJECT_HANDLE        hPrivateKey,
      CK_OBJECT_HANDLE_PTR    hCertificates,
      CK_ULONG                ulCertificateLen,
      CK_ULONG                ulFlags
);
```

Возвращаемое значение:

`CKR_OK` – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs-11v2-20.pdf, стр. 375).

Параметры:

`hSession`	*идентификатор сессии;*
`pData`	*данные для подписи;*
`ulDataLen`	*размер данных для подписи;*
`hSignCertificate`	*идентификатор объекта типа `CKO_CERTIFICATE`, указывающий на сертификат создателя сообщения;*
`pOut`	*указатель на массив байт, в который передаются данные. Память должна быть выделена заранее. Чтобы узнать необходимый размер памяти необходимо вызвать функцию, передав NULL в `pOut`. Необходимый размер будет возвращен в параметре `pulOutLen`.*
`pulOutLen`	*указатель на длину созданного буфера;*
`hPrivateKey`	*идентификатор объекта типа `CKO_PRIVATE_KEY`, указывающий на закрытый ключ создателя сообщения;*
`hCertificates`	*указатель на массив сертификатов, которые следует добавить в сообщение;*
`ulCertificateLen`	*количество сертификатов в параметре `hCertificate`;*
`ulFlags`	*флаги. Переменная `CK_ULONG` может иметь следующие значения:* *0 – исходные данные, на которые ссылается указатель `pData`, сохраняются вместе с подписанным сообщением;* *0x40 – исходные данные, на которые ссылается указатель `pData`, не сохраняются вместе с подписанным сообщением.*

C_ISBC_pkcs7Verify

```
CK_RV C_ISBC_pkcs7Verify
 (
      CK_BYTE_PTR            pPkcs7,
      CK_ULONG               ulPkcs7Len,
      CK_BYTE_PTR            pData,
      CK_ULONG               ulDataLen
);
```

Возвращаемое значение:
CKR_OK – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs-11v2-20.pdf, стр. 375).

Параметры:

pPkcs7	*указатель на байт-массив, содержащий объект данных, подписанный в формате PKCS#7;*
ulPkcs7Len	*размер объекта данных, на который ссылается указатель pPkcs7;*
pData	*указатель на массив, содержащий данные, если таковые отсутствуют непосредственно в самом объекте данных (если при создании подписи ulFlags=0x40);*
ulDataLen	*размер объекта данных, на который ссылается указатель pData.*

C_ISBC_InitToken

```
CK_RV C_ISBC_InitToken
 (
      CK_SLOT_ID              slotID,
      CK_UTF8CHAR_PTR         pPin,
      CK_ULONG                ulPinLen,
      CK_UTF8CHAR_PTR         pLabel,
      CK_BYTE                 bSoPinRc,
      CK_BYTE                 bUserPinRc
);
```

Возвращаемое значение:

`CKR_OK` – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs-11v2-20.pdf, стр. 375).

Параметры:

`slotID`	*ID слота/считывателя смарт-карт;*
`pPin`	*значение SO PIN;*
`ulPinLen`	*длина `pPin`;*
`pLabel`	*метка токена;*
`bSoPinRc`	*число попыток ввода SO PIN, значение должны быть >0 и <= 0xoF. Значения 0xoF означает, что счетчик отключен (если карта/токен это поддерживает);*
`bUserPinRc`	*число попыток ввода User PIN, значение должны быть >0 и <= 0xoF. Значения 0xoF означает, что счетчик отключен (если карта/токен это поддерживает)*

C_ISBC_GetCryptoProInfo

```
CK_RV C_ISBC_GetCryptoProInfo
(
      CK_SLOT_ID        slotID,
      CK_ULONG          *pulCount,
      CK_BYTE           *pbNames,
      CK_ULONG          *pulNamesLen
);
```

Возвращаемое значение:
`CKR_OK` – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs-11v2-20.pdf, стр. 375).

Параметры:

`slotID`	*ID слота/считывателя смарт-карт;*
`pulCount`	*количество контейнеров;*
`pbNames`	*имена контейнеров. Память должна быть выделена заранее. Чтобы узнать необходимый размер памяти необходимо вызвать функцию передав NULL в `pbNames`. Необходимый размер будет возвращен в параметре `pulNamesLen`;*
`pulNamesLen`	*размер буфера `pbNames`;*

Формат `pbNames`: последовательно тегов 0x30, содержащих тег 0x16 с именем контейнера в кодировке ISO 646 (IA5).

C_ACS_LoadCert

```
CK_RV C_ACS_LoadCert
(
CK_SESSION_HANDLE hSession,
CK_BYTE_PTR       pP12Buffer,
CK_ULONG            ulpP12BufferLen,
  CK_CHAR_PTR       pPass,
  CK_ULONG          ulPassLen
);
```

Параметры:

`hSession`	*дескриптор сессии;*
`pP12Buffer`	*получает файловый буфер файла сертификата;*
`ulpP12BufferLen`	*длина файлового буфера;*
`pPass`	*пароль к сертификату;*
`ulPassLen`	*длина пароля к сертификату в байтах;*

C_ISBC_CheckSM

```
CK_RV C_ISBC_CheckSM
(
CK_SLOT_ID  slotID,
CK_BYTE           *pbRes,


);
```

Параметры:

`slotID`	*ID слота/считывателя смарт-карт;*
`pbRes`	*Указатель на возвращаемое значение:* *0 – не поддерживается;* *1 – поддерживается, но не активирован;* *2 – поддерживается и активирован.*