РУТОКЕН®

Руководство разработчика
Высокоуровневые интерфейсы

Оглавление

Microsoft CryptoAPI 4
Реализация и использование 4
Основные свойства криптопровайдера 4
Обработка ошибок 4
Особенности работы 4
Расширение функциональности 4
Поддерживаемые параметры CryptSetProvParam 5
Поддерживаемые параметры CryptGetProvParam 7
Кэширование PIN-кода 9
Реализации стандарта PKCS#11 10
Библиотека rtPKCS11ECP 10
Библиотека rtPKCS11 10
Системные требования 10
Список функций стандарта PKCS#11 12
Список поддерживаемых функций расширения стандарта PKCS#11 14
Механизмы 15
Коды ошибок 16
Объекты 17
Объекты, поддерживаемые устройствами Рутокен 17
Объекты, поддерживаемые устройствами Рутокен ЭЦП 19
Функции стандарта PKCS#11 22
Функции расширения стандарта PKCS#11 22
Информация об использовании библиотеки rtPKCS11 28
Работа со слотами и токенами 28
Особенности работы различных типов устройств Рутокен 29
Совместимость с Рутокен CSP 30

Создание объектов из rtPKCS11 в rtCSP 30
Создание объектов из rtCSP в rtPKCS11 31
Разделение общих данных в механизме совместимости 31
Примеры использования 32

Microsoft CryptoAPI

Реализация и использование

Реализация интерфейса CryptoAPI содержится в библиотеке rtCSP.dll. Реализованный интерфейс называется Cryptographic Service Provider(CSP) или криптопровайдер. Работа с Рутокен CSP проходит так же, как и со всеми остальными криптопровайдерами (более подробно об этом см. раздел MSDN «Cryptography Functions», подраздел «Base Cryptography Functions» (http://msdn.microsoft.com/en-us/library/aa380252(VS.85).aspx#base_cryptography_functions)). При работе следует указывать имя криптопровайдера «Aktiv ruToken CSP v1.0» и тип PROV_RSA_FULL.

Основные свойства криптопровайдера

Имя	Aktiv ruToken CSP v1.0
Тип	1 (PROV_RSA_FULL)
Поддерживаемые ассиметричные алгоритмы	RSA
Поддерживаемые симметричные алгоритмы	RC2, RC4, DES, Triple DES (2 key), Triple DES(3 key)
Поддерживаемые алгоритмы хэширования	MD5, SHA1

Обработка ошибок

Все функции CryptoAPI возвращают либо TRUE, либо FALSE. Возвращённое значение FALSE означает, что работа функции завершена с ошибкой. Для получения информации об ошибке необходимо вызвать функцию GetLastError().

Особенности работы

Так как для хранения ключей используются смарткарты, то для работы с секретными данными необходимы PIN-коды. В связи с этим при использовании криптопровайдера могут быть отображены различные диалоги. Но не всегда отображение диалогов разрешено. Например, при использовании криптопровайдера из сервиса (службы), отображение диалогов не всегда возможно. Но, тем не менее, требуется доступ к защищённой PIN-кодом информации. Для разрешения таких ситуаций, при инициализации контекста криптопровайдера (CryptAcquireContext), необходимо в качестве флагов передать флаг CRYPT_SILENT – это гарантирует то, что из криптопровайдера не будет отображено ни одного диалога. Для передачи же PIN-кода есть функция CryptSetProvParam с параметром PP_SIGNATURE_PIN или PP_KEYEXCHANGE_PIN. Более подробно об этих параметрах см. MSDN.

Расширение функциональности

Для расширения возможностей криптопровайдера предусмотрено введение специфичных для криптопровайдера параметров функции CryptSetProvParam/CryptGetProvParam

Поддерживаемые параметры CryptSetProvParam

Параметры и значения	Комментарий
PP_CLIENT_HWND 1 0x1	В соответствии с MSDN
PP_DELETEKEY 24 0x18	Как MS_ENHANCED_PROV
PP_KEYEXCHANGE_ALG	Как MS_ENHANCED_PROV
PP_KEYEXCHANGE_PIN 32 0x20	Этот параметр задаёт PIN-код. Если in_pcbData NULL, то PIN-код, закэшированный в провайдере, будет сброшен
PP_KEYEXCHANGE_KEYSIZE	Как MS_ENHANCED_PROV
PP_KEYSET_SEC_DESCR 8 0x8	Как MS_ENHANCED_PROV
PP_PIN_PROMPT_STRING 44 0x2C	Как MS_ENHANCED_PROV
PP_ROOT_CERTSTORE 46 0x2E	Как MS_ENHANCED_PROV
PP_SIGNATURE_ALG	Как MS_ENHANCED_PROV
PP_SIGNATURE_PIN 33 0x21	То же что и PP_KEYEXCHANGE_PIN
PP_SIGNATURE_KEYSIZE	Как MS_ENHANCED_PROV
PP_UI_PROMPT 21 0x15	Как MS_ENHANCED_PROV
PP_USE_HARDWARE_RNG 38 0x26	Не поддерживается (для Рутокен ЭЦП используется генератор случайных чисел, встроенный в токен для операций генерации контейнерных ключевых пар, за исключением генерации архивируемой ключевой пары; и для генерации случайного числа CryptGenRandom)
PP_USER_CERTSTORE 42 0x2A	Как MS_ENHANCED_PROV
PP_SECURE_KEYEXCHANGE_PIN 47 0x2F	Как MS_ENHANCED_PROV
PP_SECURE_SIGNATURE_PIN 48 0x30	Как MS_ENHANCED_PROV
PP_SMARTCARD_READER 43 0x2B	См. PP_RTCSP_CURRENT_READER
PP_SMARTCARD_GUID 45 0x2D	Как MS_ENHANCED_PROV

Параметры и значения	Комментарий
PP_RTCSP_CURRENT_READER 65543 0x10007	Параметр расширения. Задать имя ридера, с которым необходимо работать в рамках данного контекста. При этом при запросе контейнеров (PP_ENUM_CONTAINERS) будут выведены контейнеры только для заданного ридера. Кроме того, при вызове функции CPGetProvParam с параметром PP_RTCSP_IS_PIN_SET будет возвращена информация для заданного таким образом ридера. Параметр in_pcbtData должен содержать имя ридера в ANSI представлении.
PP_RTCSP_SET_DEF_CONT 65545 0x10009	Параметр расширения. Задать контейнер по умолчанию для ридера, над которым был открыт контекст. Параметр in_pcbtData должен содержать указатель на DWORD, по которому лежит значение TRUE для выставления контейнера криптопровайдера контейнером по умолчанию. Если же там значение FALSE, то контейнером по умолчанию будет выставлен контейнер с индексом 0 в списке контейнеров токена.

Поддерживаемые параметры CryptGetProvParam

Параметры и значения	Комментарий
PP_ADMIN_PIN 31 0x1F	Как MS_ENHANCED_PROV
PP_APPLI_CERT 18 0x12	Как MS_ENHANCED_PROV
PP_CHANGE_PASSWORD 7 0x7	Как MS_ENHANCED_PROV
PP_CERTCHAIN 9 0x9	Как MS_ENHANCED_PROV
PP_CONTAINER 6 0x6	В соответствии с MSDN
PP_CRYPT_COUNT_KEY_USE 41 0x29	Как MS_ENHANCED_PROV
PP_ENUMALGS 1 0x1	В соответствии с MSDN.
PP_ENUMALGS_EX 22 0x16	В соответствии с MSDN.
PP_ENUMCONTAINERS 2 0x2	В соответствии с MSDN
PP_ENUMELECTROOTS 26 0x1A	Как MS_ENHANCED_PROV
PP_ENUMEX_SIGNING_PROT 40 0x28	Как MS_ENHANCED_PROV
PP_ENUMMANDROOTS 25 0x19	Как MS_ENHANCED_PROV
PP_IMPTYPE 3 0x3	В соответствии с MSDN (CRYPT_IMPL_MIXED | CRYPT_IMPL_REMOVABLE)
PP_KEY_TYPE_SUBTYPE 10 0xA	Как MS_ENHANCED_PROV
PP_KEYEXCHANGE_PIN 32 0x20	Как MS_ENHANCED_PROV

PP_KEYSET_SEC_DESCR 8 0x8	Как MS_ENHANCED_PROV
PP_KEYSET_TYPE 27 0x1B	Как MS_ENHANCED_PROV
PP_KEYSPEC 39 0x27	Как MS_ENHANCED_PROV
PP_KEYSTORAGE 17 0x11	Как MS_ENHANCED_PROV
PP_KEYX_KEYSIZE_INC 35 0x23	Как MS_ENHANCED_PROV
PP_NAME 4 0x4	В соответствии с MSDN
PP_PROVTYPE 16 0x10	В соответствии с MSDN
PP_ROOT_CERTSTORE 46 0x2E	Как MS_ENHANCED_PROV
PP_SESSION_KEYSIZE 20 0x14	Как MS_ENHANCED_PROV
PP_SGC_INFO 37 0x25	Как MS_ENHANCED_PROV
PP_SIG_KEYSIZE_INC 34 0x22	Как MS_ENHANCED_PROV
PP_SIGNATURE_PIN 33 0x21	Как MS_ENHANCED_PROV
PP_SMARTCARD_GUID 45 0x2D	Как MS_ENHANCED_PROV
PP_SMARTCARD_READER 43 0x2B	Как MS_ENHANCED_PROV
PP_SYM_KEYSIZE 19 0x13	Как MS_ENHANCED_PROV
PP_UI_PROMPT 21 0x15	Как MS_ENHANCED_PROV
PP_UNIQUE_CONTAINER 36 0x24	В соответствии с MSDN

PP_USE_HARDWARE_RNG 38 0x26	Как MS_ENHANCED_PROV
PP_USER_CERTSTORE 42 0x2A	Как MS_ENHANCED_PROV
PP_VERSION 5 0x5	В соответствии с MSDN
PP_RTCSP_IS_PIN_SET 65546 0x1000A	Параметр расширения. Проверить, был ли выставлен PIN-код для ридера. Если контекст открыт над ридером, то проверка осуществляется для ридера, над которым открыт контекст. Если контекст открыт не над ридером, то проверка осуществляется для ридера, заданного при помощи функции CPSetProvParam с параметром PP_RTCSP_CURRENT_READER. PIN-код выставляется функцией CPSetPRovParam с параметром PP_SIGNATURE_PIN или PP_KEYEXCHANGE_PIN

Кэширование PIN-кода

Для получения доступа к закрытым данным без отображения диалогов в CryptoAPI предусмотрен механизм задания PIN-кода через функцию CrytpSetProcParam с параметрами PP_SIGNATURE_PIN и PP_KEYEXCHANGE_PIN (более подробно см. MSDN). В криптопровайдере Рутокен эти два параметра не различимы и играют одинаковую роль – задают PIN-код для работы с токеном.
После выставления PIN-кода, он сохраняется в рамках сессии данного процесса. Заданный PIN-код для ридера через контекст криптопорвайдера доступен для всех контекстов криптопровайера в рамках той же сессии того же процесса, даже после закрытия контекста, через который он был выставлен.

Заданный PIN-код будет сброшен при выполнении одного из следующих условий:

1. Выгрузка библиотеки
2. Отключение ридера
3. Вызов функции CryptSetProvParam с параметром PP_SIGNATURE_PIN или PP_KEYEXCHANGE_PIN и pbData = NULL.

Реализации стандарта PKCS#11

Стандарт PKCS#11 разработан компанией RSA Laboratories, более подробно с текстом стандарта можно ознакомиться на странице http://www.rsa.com/rsalabs/node.asp?id=2133. Библиотека rtPKCS11 реализует версию стандарта 2.20.

Библиотека rtPKCS11ECP

Библиотека rtPKCS11ECP предназначена для работы с электронными идентификаторами Рутокен ЭЦП. В ней реализована функциональность, относящаяся к реализации российских криптографических алгоритмов (ГОСТ 34.10-2001, ГОСТ 34.11-94, ГОСТ 28147-89).

Библиотека rtPKCS11ECP входит в состав драйверов Рутокен, но может быть использована и без собственно драйверов, поскольку Рутокен ЭЦП является CCID-совместимым устройством. Драйверы CCID входят в состав современных операционных систем.

Библиотека rtPKCS11ECP выполнена в виде dll-модуля, написанного на языке C++.

Библиотека rtPKCS11ECP сохраняет обратную совместимость по формату данных со всеми предыдущими версиями, т.е. объекты, созданные с помощью прежних версий библиотеки можно продолжать использовать.

Библиотека rtPKCS11

Библиотека rtPKCS11 позволяет работать с электронными идентификаторами Рутокен и Рутокен ЭЦП по стандарту PKCS#11. Эта библиотека реализует функциональность, связанную с использованием зарубежных криптографических алгоритмов (RSA).

Библиотека rtPKCS11 входит в состав драйверов Рутокен, это позволяет конечным пользователям получить актуальные обновления и исправления в библиотеке максимально просто и быстро. Установка и удаление библиотеки происходит автоматически вместе драйверами Рутокен. Библиотека rtPKCS11 выполнена в виде dll-модуля, написанного на языке C++.

Библиотека rtPKCS11 сохраняет обратную совместимость по формату данных со всеми предыдущими версиями, т.е. объекты, созданные с помощью прежних версий библиотеки можно продолжать использовать.

Системные требования

Поддерживаются операционные системы Microsoft Windows в соответствии с системными требованиями драйверов Рутокен.

С драйверами Рутокен для платформы Win32 распространяется библиотека rtPKCS11 в одной конфигурации: “Unicode Win32”. С драйверами Рутокен для платформы x64 распространяется библиотека rtPKCS11 в двух конфигурациях: “Unicode Win32” и “Unicode x64”.

Работа библиотеки rtPKCS11ECP не зависит от других модулей.

Работа библиотеки rtPKCS11 зависит от следующих модулей: rtLib, rtAPIi, rtGrTools, rtCSP. Указанные модули распространяется в составе драйверов Рутокен, их установка, обновление, удаление происходит автоматически в процессе работы программы установки драйверов Рутокен.

Список функций стандарта PKCS#11

Библиотека rtPKCS#11 реализует необходимые для работы всех устройств Рутокен функции стандарта PKCS#11. Функции, не реализованные в библиотеке rtPKCS11, при вызове возвращают код ошибки CKR_FUNCTION_NOT_SUPPORTED.

В таблице приведен список всех функций стандарта PKCS#11, функции работу которых поддерживают устройства Рутокен отмечены знаком «**+**».

Функции стандарта PKCS#11.	**rtPKCS11**		**rtPKCS11ECP**
	Рутокен	**Рутокен ЭЦП**	**Рутокен ЭЦП**
C_Initialize	**+**	**+**	**+**
C_Finalize	**+**	**+**	**+**
C_GetInfo	**+**	**+**	**+**
C_GetFunctionList	**+**	**+**	**+**
C_GetSlotList	**+**	**+**	**+**
C_GetSlotInfo	**+**	**+**	**+**
C_GetTokenInfo	**+**	**+**	**+**
C_GetMechanismList	**+**	**+**	**+**
C_GetMechanismInfo	**+**	**+**	**+**
C_InitToken	**+**	**+**	**+**
C_InitPIN	**+**	**+**	**+**
C_SetPIN	**+**	**+**	**+**
C_OpenSession	**+**	**+**	**+**
C_CloseSession	**+**	**+**	**+**
C_CloseAllSessions	**+**	**+**	**+**
C_GetSessionInfo	**+**	**+**	**+**
C_GetOperationState			
C_SetOperationState			
C_Login	**+**	**+**	**+**
C_Logout	**+**	**+**	**+**
C_CreateObject	**+**	**+**	**+**
C_CopyObject			
C_DestroyObject	**+**	**+**	**+**
C_GetObjectSize			
C_GetAttributeValue	**+**	**+**	**+**
C_SetAttributeValue	**+**	**+**	**+**
C_FindObjectsInit	**+**	**+**	**+**
C_FindObjects	**+**	**+**	**+**
C_FindObjectsFinal	**+**	**+**	**+**
C_EncryptInit	**+**	**+**	**+**
C_Encrypt	**+**	**+**	**+**
C_EncryptUpdate			
C_EncryptFinal			

Функции стандарта PKCS#11.	rtPKCS11		rtPKCS11ECP
	Рутокен	Рутокен ЭЦП	Рутокен ЭЦП
C_DecryptInit	+	+	+
C_Decrypt	+	+	+
C_DecryptUpdate			
C_DecryptFinal			
C_DigestInit	+	+	+
C_Digest	+	+	+
C_DigestUpdate			+
C_DigestKey			
C_DigestFinal			+
C_SignInit	+	+	+
C_Sign	+	+	+
C_SignUpdate			
C_SignFinal			
C_SignRecoverInit			
C_SignRecover			
C_VerifyInit	+	+	+
C_Verify	+	+	+
C_VerifyUpdate			
C_VerifyFinal			
C_VerifyRecoverInit			
C_VerifyRecover			
C_DigestEncryptUpdate			
C_DecryptDigestUpdate			
C_SignEncryptUpdate			
C_DecryptVerifyUpdate			
C_GenerateKey	+	+	+
C_GenerateKeyPair	+	+	+
C_WrapKey			+
C_UnwrapKey			+
C_DeriveKey			+
C_SeedRandom			
C_GenerateRandom	+	+	+
C_GetFunctionStatus			
C_CancelFunction			
C_WaitForSlotEvent	+	+	+

Список поддерживаемых функций расширения стандарта PKCS#11

Библиотека rtPKCS#11 реализует функции расширения стандарта PKCS#11 для поддержки специфической функциональности устройств Рутокен. Функции расширения стандарта не поддерживающие работу с каким лидо типом устройств Рутокен, при вызове возвращают код ошибки CKR_FUNCTION_NOT_SUPPORTED.

В Таблица 1 приведен список всех функций расширения стандарта PKCS#11, функции работу, которых поддерживают различные типы устройства Рутокен отмечены знаком «+».

Таблица 1.

Функции расширения стандарта PKCS#11.	**rtPKCS11**		**rtPKCS11ECP**
	Рутокен	**Рутокен ЭЦП**	**Рутокен ЭЦП**
C_EX_GetFunctionListExtended	**+**	**+**	**+**
C_EX_GetTokenInfoExtended	**+**	**+**	**+**
C_EX_InitToken	**+**	**+**	**+**
C_EX_UnblockUserPIN	**+**	**+**	**+**
C_EX_SetTokenName	**+**	**+**	**+**
C_EX_GetTokenName	**+**	**+**	**+**
C_EX_SetLicense		**+**	**+**
C_EX_GetLicense		**+**	**+**

Механизмы

Библиотека rtPKCS11 поддерживает работу устройств Рутокен со следующими механизмами:

Подерживаемые механизмы PKCS#11.	**rtPKCS11**		**rtPKCS11ECP**
	Рутокен	**Рутокен ЭЦП**	**Рутокен ЭЦП**
CKM_RSA_PKCS_KEY_PAIR_GEN	**+**	**+**	
CKM_RSA_PKCS	**+**	**+**	
CKM_MD2	**+**		
CKM_MD5	**+**	**+**	
CKM_SHA_1	**+**	**+**	
CKM_GENERIC_SECRET_KEY_GEN	**+**	**+**	**-**
CKM_GOSTR3410_KEY_PAIR_GEN			**+**
CKM_GOSTR3410			**+**
CKM_GOSTR3410_WITH_GOSTR3411			**+/-** **(не реализовано для C_Verify*)**
CKM_GOSTR3410_KEY_WRAP			**-**
CKM_GOSTR3410_DERIVE			**+**
CKM_GOSTR3411			**+**
CKM_GOSTR3411_HMAC			**-**
CKM_GOST28147_KEY_GEN			**+**
CKM_GOST28147_ECB			**+**
CKM_GOST28147			**+**
CKM_GOST28147_MAC			**+**
CKM_GOST28147_KEY_WRAP			**+**
CKM_GOST_KEY_GEN	**+**		**-**
CKM_GOST	**+**		**-**

Коды ошибок

Стандартные коды ошибок – эти коды ошибок описаны в стандарте. Эти коды ошибок могут возвращать стандартные функции и функции расширения.

В силу особенностей реализации библиотеки rtPKCS11, некоторые стандартные функции могут вернуть стандартный код ошибки PKCS#11 не входящий в список допустимых для данной функции. Подобная ситуация является исключением. Стандартные коды ошибок возвращаемые каждой функцией в исключительных ситуациях перечислены в описании для каждой функции отдельно.

Расширенные коды ошибок - эти коды ошибок определены специально для библиотеки rtPKCS11. Эти коды ошибок могут возвращать стандартные функции и функции расширения. Ниже приведен список специальных кодов ошибок.

CKR_CORRUPTED_MAPFILE – данная ошибка возвращается при повреждении MAP-файла.

CKR_RTPKCS11_DATA_CORRUPTED - данная ошибка возвращается, если было обнаружено нарушение целостности данных на токене.

CKR_WRONG_VERSION_FIELD - данная ошибка возвращается, если файл содержащий объект PKCS#11 имеет не корректную версию.

CKR_RTPKCS11_RSF_DATA_CORRUPTED – данная ошибка возвращается, если попытка использовать RSF-файл завершилась неудачей.

Объекты

Объекты, поддерживаемые устройствами Рутокен

Стандартные объекты PKCS#11

Библиотека rtPKCS11 поддерживает работу устройств Рутокен со следующими классами объектов определённых в стандарте PKCS#11:

CKO_DATA
CKO_PUBLIC_KEY
    тип CKK_RSA
CKO_PRIVATE_KEY
    тип CKK_RSA
CKO_SECRET_KEY
    тип CKK_GENERIC_SECRET
CKO_CERTIFICATE
    тип CKC_X_509

При работе с объектами поддерживаются следующие стандартные атрибуты. Под поддержкой атрибута понимается, какое либо использование атрибута – задание при создании, установка после создания, чтение, поиск по атрибуту или какое либо другое использование. Для каждого конкретного атрибута – наиболее подробная информация о использовании описана в стандарте.

Таблица 2.

Стандартные атрибуты объектов.	**Класса объектов.**				
	CKO_DATA	CKO_PUBLIC_KEY	CKO_PRIVATE_KEY	CKO_SECRET_KEY	CKO_CERTIFICATE
CKA_CLASS	+	+	+	+	+
CKA_TOKEN	+	+	+	+	+
CKA_PRIVATE	+	+	+	+	+
CKA_MODIFIABLE	+	+	+	+	+
CKA_LABEL	+	+	+	+	+
CKA_COPYABLE	+	+	+	+	+
CKA_APPLICATION	+				
CKA_VALUE	+			+	+
CKA_VALUE_LEN				+	
CKA_OBJECT_ID	+				
CKA_KEY_TYPE		+	+	+	
CKA_ID		+	+	+	+
CKA_START_DATE		+	+	+	+
CKA_END_DATE		+	+	+	+
CKA_DERIVE		+	+	+	
CKA_LOCAL		+	+	+	
CKA_KEY_GEN_MECHANISM		+	+	+	
CKA_ALLOWED_MECHANISMS		+	+	+	

Стандартные атрибуты объектов.	**Класса объектов.**				
	CKO_DATA	CKO_PUBLIC_KEY	CKO_PRIVATE_KEY	CKO_SECRET_KEY	CKO_CERTIFICATE
CKA_SUBJECT		+	+		+
CKA_ENCRYPT		+		+	
CKA_VERIFY		+		+	
CKA_VERIFY_RECOVER		+			
CKA_WRAP		+		+	
CKA_TRUSTED		+		+	+
CKA_SENSITIVE				+	
CKA_DECRYPT			+	+	
CKA_SIGN			+	+	
CKA_SIGN_RECOVER			+		
CKA_UNWRAP			+	+	
CKA_EXTRACTABLE			+	+	
CKA_NEVER_EXTRACTABLE			+	+	
CKA_ALWAYS_SENSITIVE			+	+	
CKA_ALWAYS_AUTHENTICATE			+		
CKA_WRAP_WITH_TRUSTED			+	+	
CKA_UNWRAP_TEMPLATE			+	+	
CKA_WRAP_TEMPLATE		+		+	
CKA_MODULUS_BITS		+			
CKA_MODULUS		+	+		
CKA_PUBLIC_EXPONENT		+	+		
CKA_PRIVATE_EXPONENT			+		
CKA_PRIME_1			+		
CKA_PRIME_2			+		
CKA_EXPONENT_1			+		
CKA_EXPONENT_2			+		
CKA_COEFFICIENT			+		
CKA_CERTIFICATE_TYPE					+
CKA_CERTIFICATE_CATEGORY					+
CKA_CHECK_VALUE				+	+
CKA_ISSUER					+
CKA_SERIAL_NUMBER					+
CKA_JAVA_MIDP_SECURITY_DOMAIN					+
CKA_URL					+
CKA_HASH_OF_SUBJECT_PUBLIC_KEY					+
CKA_HASH_OF_ISSUER_PUBLIC_KEY					+
CKA_NAME_HASH_ALGORITHM					+

Объекты расширения стандарта PKCS#11

Библиотека rtPKCS11 поддерживает работу устройств Рутокен со следующими типами объектов расширения стандарта PKCS#11:

CKO_SECRET_KEY
    тип CKK_GOST

В дополнение к поддерживаемым атрибутам для класса CKO_SECRET_KEY, объекты CKO_SECRET_KEY типа CKK_GOST поддерживают атрибуты расширения стандарта PKCS#11.

Таблица 3

Атрибуты расширения стандарта PKCS#11.	**Класс объекта**
	CKO_SECRET_KEY, Тип CKK_GOST
CKA_GOST_KEY_OPTIONS	+
CKA_GOST_KEY_FLAGS	+

Объекты, поддерживаемые устройствами Рутокен ЭЦП

Стандартные объекты PKCS#11

Библиотека rtPKCS11 поддерживает работу устройств Рутокен ЭЦП со следующими классами объектов определённых в стандарте PKCS#11:

CKO_DATA
CKO_PUBLIC_KEY
    тип CKK_RSA
CKO_PRIVATE_KEY
    тип CKK_RSA
CKO_SECRET_KEY
    тип CKK_GENERIC_SECRET
CKO_CERTIFICATE
    тип CKC_X_509

При работе с объектами поддерживаются следующие стандартные атрибуты. Под поддержкой атрибута понимается, какое либо использование атрибута – задание при создании, установка после создания, чтение, поиск по атрибуту или какое либо другое использование. Для каждого конкретного атрибута – наиболее подробная информация об использовании описана в стандарте.

Стандартные атрибуты объектов.	**Классы объектов.**				
	CKO_DATA	CKO_PUBLIC_KEY	CKO_PRIVATE_KEY	CKO_SECRET_KEY	CKO_CERTIFICATE
CKA_CLASS	+	+	+	+	+
CKA_TOKEN	+	+	+	+	+

Стандартные атрибуты объектов.	**Классы объектов.**				
	CKO_DATA	CKO_PUBLIC_KEY	CKO_PRIVATE_KEY	CKO_SECRET_KEY	CKO_CERTIFICATE
CKA_PRIVATE	+	+	+	+	+
CKA_MODIFIABLE	+	+	+	+	+
CKA_LABEL	+	+	+	+	+
CKA_COPYABLE	+	+	+	+	+
CKA_APPLICATION	+				
CKA_VALUE	+			+	+
CKA_VALUE_LEN				+	
CKA_OBJECT_ID	+				
CKA_KEY_TYPE		+	+	+	
CKA_ID		+	+	+	+
CKA_START_DATE		+	+	+	+
CKA_END_DATE		+	+	+	+
CKA_DERIVE		+	+	+	
CKA_LOCAL		+	+	+	
CKA_KEY_GEN_MECHANISM		+	+	+	
CKA_ALLOWED_MECHANISMS		+	+	+	
CKA_SUBJECT		+	+		+
CKA_ENCRYPT		+		+	
CKA_VERIFY		+		+	
CKA_VERIFY_RECOVER		+			
CKA_WRAP		+		+	
CKA_TRUSTED		+		+	+
CKA_SENSITIVE				+	
CKA_DECRYPT			+	+	
CKA_SIGN			+	+	
CKA_SIGN_RECOVER			+		
CKA_UNWRAP			+	+	
CKA_EXTRACTABLE			+	+	
CKA_NEVER_EXTRACTABLE			+	+	
CKA_ALWAYS_SENSITIVE			+	+	
CKA_ALWAYS_AUTHENTICATE			+		
CKA_WRAP_WITH_TRUSTED			+	+	
CKA_UNWRAP_TEMPLATE			+	+	
CKA_WRAP_TEMPLATE		+		+	
CKA_MODULUS_BITS		+			
CKA_MODULUS		+	+		
CKA_PUBLIC_EXPONENT		+	+		
CKA_PRIVATE_EXPONENT			+		
CKA_PRIME_1			+		
CKA_PRIME_2			+		
CKA_EXPONENT_1			+		
CKA_EXPONENT_2			+		
CKA_COEFFICIENT			+		
CKA_CERTIFICATE_TYPE					+

Стандартные атрибуты объектов.	**Классы объектов.**				
	CKO_DATA	CKO_PUBLIC_KEY	CKO_PRIVATE_KEY	CKO_SECRET_KEY	CKO_CERTIFICATE
CKA_CERTIFICATE_CATEGORY					+
CKA_CHECK_VALUE				+	+
CKA_ISSUER					+
CKA_SERIAL_NUMBER					+
CKA_JAVA_MIDP_SECURITY_DOMAIN					+
CKA_URL					+
CKA_HASH_OF_SUBJECT_PUBLIC_KEY					+
CKA_HASH_OF_ISSUER_PUBLIC_KEY					+
CKA_NAME_HASH_ALGORITHM					+

Функции стандарта PKCS#11

Наиболее полное описание стандартных функций PKCS#11 приведено в описании самого стандарта. (pkcs-11v2-20.pdf). Поэтому в данном документе описание функций не приводится.

Функции расширения стандарта PKCS#11

Функции расширения предоставляют разработчику дополнительные возможности по работе с Рутокен, такие как:

- получение расширенной информации о токене
- разблокировка PIN-кода пользователя
- задание и считывание имени токена произвольной длины
- запись и чтение информации для лицензирования приложений

C_EX_GetTokenInfoExtended

Функция позволяет получить расширенную информацию о токене. Расширенная информация о токене возвращается в структуре типа CK_TOKEN_INFO_EXTENDED.

Синтаксис

```
CK_DEFINE_FUNCTION(CK_RV, C_EX_GetTokenInfoExtended)
(
      CK_SLOT_ID                    slotID,
      CK_TOKEN_INFO_EXTENDED_PTR    pInfo
);
```

Параметры

slotID

[in] ID слота с подключенным токеном, о котором необходимо получить информацию.

pInfo

[out] указатель на структуру CK_TOKEN_INFO_EXTENDED.

Возвращаемые значения

Стандартные коды ошибок:
CKR_CRYPTOKI_NOT_INITIALIZED,
CKR_DEVICE_ERROR,
CKR_DEVICE_MEMORY,
CKR_DEVICE_REMOVED,
CKR_FUNCTION_FAILED,
CKR_GENERAL_ERROR,
CKR_HOST_MEMORY,
CKR_OK,
CKR_SLOT_ID_INVALID,
CKR_TOKEN_NOT_PRESENT,
CKR_TOKEN_NOT_RECOGNIZED,
CKR_ARGUMENTS_BAD
Расширенные коды ошибок.

Примечание

Функция позволяет получить расширенную информацию о токене. Расширенная информация о токене возвращается в структуре типа CK_TOKEN_INFO_EXTENDED.

```
typedef struct _CK_TOKEN_INFO_EXTENDED {
      CK_ULONG    ulSizeofThisStructure;
      CK_ULONG    ulTokenType;
      CK_ULONG    ulProtocolNumber;
      CK_ULONG    ulMicrocodeNumber;
      CK_ULONG    ulOrderNumber;
      CK_FLAGS    flags;
      CK_ULONG    ulMaxAdminPinLen;
      CK_ULONG    ulMinAdminPinLen;
      CK_ULONG    ulMaxUserPinLen;
      CK_ULONG    ulMinUserPinLen;
      CK_ULONG    ulMaxAdminRetryCount;
      CK_ULONG    ulAdminRetryCountLeft;
      CK_ULONG    ulMaxUserRetryCount;
      CK_ULONG    ulUserRetryCountLeft;
      CK_BYTE     serialNumber[8];
      CK_ULONG    ulTotalMemory;
      CK_ULONG    ulFreeMemory;
      CK_BYTE     ATR[64];
      CK_ULONG    ulATRLen;
} CK_TOKEN_INFO_EXTENDED;
```

Основное отличие функции C_EX_GetTokenInfoExtended от сходной по назначению функции стандарта C_GetTokenInfo, в том, что функция расширения предоставляет более полную информацию о токене.

C_EX_UnblockUserPIN

Функция позволяет разблокировать PIN-код пользователя. Функция может быть вызвана только из сессии имеющей статус CKS_RW_SO_FUNCTIONS.

Синтаксис

```
CK_DEFINE_FUNCTION(CK_RV, C_EX_UnblockUserPIN)
(
      CK_SESSION_HANDLE  hSession
);
```

Параметры

hSession
: [in] идентификатор (handle) сессии.

Возвращаемые значения

Стандартные коды ошибок:
CKR_CRYPTOKI_NOT_INITIALIZED,
CKR_DEVICE_ERROR,
CKR_DEVICE_MEMORY,
CKR_DEVICE_REMOVED,

CKR_FUNCTION_CANCELED,
CKR_FUNCTION_FAILED,
CKR_GENERAL_ERROR,
CKR_HOST_MEMORY,
CKR_OK,
CKR_PIN_INVALID,
CKR_PIN_LEN_RANGE,
CKR_SESSION_CLOSED,
CKR_SESSION_READ_ONLY,
CKR_SESSION_HANDLE_INVALID,
CKR_TOKEN_WRITE_PROTECTED,
CKR_USER_NOT_LOGGED_IN,
CKR_ARGUMENTS_BAD
Расширенные коды ошибок.

Примечание

Функция позволяет сбросить счетчик неверных попыток ввода PIN-кода для CKU_USER, для успешного выполнения функции C_EX_UnblockUserPIN необходимо аутентифицироваться с правами CKU_SO.

C_EX_SetTokenName

Функция позволяет задать метку токена произвольной длины. Функция может быть вызвана только из сессии имеющей статус CKS_RW_USER_FUNCTIONS.

Синтаксис

```
CK_DEFINE_FUNCTION(CK_RV, C_EX_SetTokenName)
(
      CK_SESSION_HANDLE       hSession,
      CK_BYTE_PTR             pLabel,
      CK_ULONG                ulLabelLen
);
```

Параметры

hSession
: [in] идентификатор (handle) сессии.

pLabel
: [in] указатель на буфер содержащий метку токена.

ulLabelLen
: [in] размер буфера с меткой токена, в байтах.

Возвращаемые значения

Стандартные коды ошибок:
CKR_ARGUMENTS_BAD,
CKR_CRYPTOKI_NOT_INITIALIZED,
CKR_DEVICE_ERROR,
CKR_DEVICE_MEMORY,
CKR_DEVICE_REMOVED,
CKR_FUNCTION_FAILED,
CKR_GENERAL_ERROR,

CKR_HOST_MEMORY,
CKR_OK,
CKR_PIN_EXPIRED,
CKR_SESSION_CLOSED,
CKR_SESSION_HANDLE_INVALID,
CKR_SESSION_READ_ONLY,
CKR_TOKEN_WRITE_PROTECTED,
CKR_USER_NOT_LOGGED_IN
Расширенные коды ошибок.

Примечание

Функция позволяет установить метку токена произвольной длины, в отличии от стандартной функии C_InitToken, которая позволяет устанавливать метку токена фиксированного размера.

C_EX_GetTokenName

Функция позволяет получить метку токена произвольной длины. Функция может быть вызвана из любой сессии.

Синтаксис

```
CK_DEFINE_FUNCTION(CK_RV, C_EX_GetTokenName)
(
      CK_SESSION_HANDLE       hSession,
      CK_BYTE_PTR             pLabel,
      CK_ULONG_PTR            pulLabelLen
);
```

Параметры

hSession
: [in] идентификатор (handle) сессии.

pLabel
: [in, out] указатель на буфер содержащий метку токена.

ulLabelLen
: [in, out] размер буфера с меткой токена, в байтах.

Возвращаемые значения

Стандартные коды ошибок:
CKR_ARGUMENTS_BAD,
CKR_CRYPTOKI_NOT_INITIALIZED,
CKR_DEVICE_ERROR,
CKR_DEVICE_MEMORY,
CKR_DEVICE_REMOVED,
CKR_FUNCTION_FAILED,
CKR_GENERAL_ERROR,
CKR_HOST_MEMORY,
CKR_OK,
CKR_SESSION_CLOSED,
CKR_SESSION_HANDLE_INVALID,
CKR_BUFFER_TOO_SMALL
Расширенные коды ошибок.

Примечание

Функция позволяет получить метку токена произвольной длины, в отличии от стандартной функии C_GetTokenInfo, которая позволяет получить метку токена фиксированного размера.

Для определения необходимого количества памяти для хранения метки токена, необходимо вызвать функцию C_EX_GetTokenName с параметром pLable равным NULL_PTR, тогда в параметре pulLabelLen будет возвращен указатель на переменную с требуемым размером буфера.

C_EX_SetLicense

Функция позволяет записать лицензию на токен.

Синтаксис

```
CK_DEFINE_FUNCTION(CK_RV, C_EX_SetLicense)
(
      CK_SESSION_HANDLE       hSession,
      CK_UKONG                ulLicenseNum,
      CK_BYTE_PTR             pLicense,
      CK_ULONG                ulLicenseLen
);
```

Параметры

hSession

[in] идентификатор (handle) сессии.

ulLicenseNum

[in] идентификатор лицензии. Параметр может принимать одно из двух значений: 1 или 2.

pLicense

[in] указатель на буфер содержащий лицензию.

ulLicenseLen

[in] размер буфера с лицензией, в байтах. Параметр может принимать только одно значение: 72.

Возвращаемые значения

Стандартные коды ошибок:
CKR_ARGUMENTS_BAD,
CKR_CRYPTOKI_NOT_INITIALIZED,
CKR_DEVICE_ERROR,
CKR_DEVICE_MEMORY,
CKR_DEVICE_REMOVED,
CKR_FUNCTION_FAILED,
CKR_GENERAL_ERROR,
CKR_HOST_MEMORY,
CKR_OK,
CKR_PIN_EXPIRED,
CKR_SESSION_CLOSED,
CKR_SESSION_HANDLE_INVALID,
CKR_SESSION_READ_ONLY,
CKR_TOKEN_WRITE_PROTECTED,
CKR_USER_NOT_LOGGED_IN,
CKR_FUNCTION_NOT_SUPPORTED

Расширенные коды ошибок.

Примечание

Функция позволяет изменить лицензию на токене. Функция может быть вызвана только из сессии имеющей статус CKS_RW_USER_FUNCTIONS или CKS_RW_SO_FUNCTIONS.

При инициализации памяти токена с помощью функций C_InitToken или C_EX_InitToken ранее записанная лицензия не будет удалена.

C_EX_GetLicense

Функция позволяет прочитать лицензию, записанную на токен.

Синтаксис

```
CK_DEFINE_FUNCTION(CK_RV, C_EX_GetLicense)
(
      CK_SESSION_HANDLE       hSession,
      CK_UKONG                ulLicenseNum,
      CK_BYTE_PTR             pLicense,
      CK_ULONG_PTR            pulLicenseLen);
```

Параметры

hSession

[in] идентификатор (handle) сессии.

ulLicenseNum

[in] идентификатор лицензии. Параметр может принимать одно из двух значений: 1 или 2.

pLicense

[in, out] указатель на буфер содержащий лицензию.

pulLicenseLen

[in, out] указатель на размер буфера с лицензией, в байтах. Параметр может принимать только одно значение: 72.

Возвращаемые значения

Стандартные коды ошибок:
CKR_ARGUMENTS_BAD,
CKR_CRYPTOKI_NOT_INITIALIZED,
CKR_DEVICE_ERROR,
CKR_DEVICE_MEMORY,
CKR_DEVICE_REMOVED,
CKR_FUNCTION_FAILED,
CKR_GENERAL_ERROR,
CKR_HOST_MEMORY,
CKR_OK,
CKR_SESSION_CLOSED,
CKR_SESSION_HANDLE_INVALID,
CKR_FUNCTION_NOT_SUPPORTED
Расширенные коды ошибок.

Примечание

Функция позволяет прочитать лицензию с токена. Функция может быть вызвана только из любой сессии. Параметр ulLicenseNum может принимать значение 1 или 2. Параметр pLicense – указатель на буффер для получения лицензии. Параметр pulLicenseLen – указатель на длину буфера, длина лицензии может иметь единственное значение 72 байта, либо при передаче pLicense равного NULL_PTR, в параметре pulLicenseLen возвращается указатель на длину буфера для требуемой лицензии.

Информация об использовании библиотеки rtPKCS11

Работа со слотами и токенами

Слот в библиотеке rtPKCS11 являются универсальными, к каждому слоту в разное время может быть подключено любое устройство Рутокен или Рутокен ЭЦП.

После инициализации библиотеки вызовом функции C_Initialize, создаются слоты. Если не подключено ни одного устройства Рутокен, то в системе создается три пустых слота. После отключения устройств слоты сохраняются в системе, но помечается пустым. В системе все ранее созданные слоты будут зарегистрированы до вызова функции C_Finalize.

Как определить подключено ли к слоту устройство Рутокен?

Для определения подключен ли к конкретному слоту токен, необходимо вызвать функцию C_GetSlotInfo и проверить выставлен флаг CKF_TOKEN_PRESENT в поле flags в структуре типа CK_TOKEN_INFO.

Если флаг выставлен, то устройство Рутокен подключено, если флаг сброшен, то, к слоту не подключено ни какое устройство.

Как определить тип подключенного к слоту устройства Рутокен?

Способ 1

Для получения типа устройства необходимо вызвать функцию C_EX_GetTokenInfoExtended. В структуре типа CK_TOKEN_INFO_EXTENDED, в поле ulTokenClass будет возвращена константа, по которой можно определить тип токена:

TOKEN_CLASS_S – означает, что подключен Рутокен.
TOKEN_CLASS_ECP – означает, что подключен Рутокен ЭЦП.

Способ 2

Для получения типа устройства подключенного к конкретному слоту необходимо сравнить списки механизмов, которые возвращает функция C_GetMechanismList для данного слота:

Если к слоту подключен Рутокен, то список механизмов будет:

CKM_RSA_PKCS_KEY_PAIR_GEN
CKM_RSA_PKCS
CKM_MD2
CKM_MD5
CKM_SHA_1
CKM_GENERIC_SECRET_KEY_GEN
CKM_GOST_KEY_GEN
CKM_GOST

Если к слоту подключен Рутокен ЭЦП, то список механизмов будет:

CKM_RSA_PKCS_KEY_PAIR_GEN
CKM_RSA_PKCS
CKM_MD5
CKM_SHA_1
CKM_GENERIC_SECRET_KEY_GEN

Особенности работы различных типов устройств Рутокен

Генерация случайных чисел.

Все устройства Рутокен поддерживают аппаратную генерацию случайных данных.

Сгенерированные случайные данные используются:

1. Для заполнения теля ключа CKK_GENERIC_SECRET, сгенерированного функцией C_GenerateKey с помощью механизма CKM_GENERIC_SECRET_KEY_GEN.
2. В функции C_GenerateRandom - для заполнения буфера переданного пользователем.

Генерация ключевых пар RSA.

1. Устройства Рутокен не поддерживают аппаратную генерацию ключевых пар RSA, для генерации используется криптопровайдер "Microsoft Enhanced Cryptographic Provider v1.0".
2. Устройство Рутокен ЭЦП поддерживает аппаратную генерацию ключевых пар RSA, генерация происходит в памяти Рутокен ЭЦП, ключевая пара никогда не покидает памяти устройства.

Генерация ключей CKK_GOST.

1. Устройства Рутокен поддерживают аппаратную генерацию ключей CKK_GOST по алгоритму ГОСТ 28147-89, генерация ключей происходит в памяти устройства, сгенерированные ключи никогда не покидают устройство Рутокен.
2. Устройство Рутокен ЭЦП не поддерживает генерацию ключей CKK_GOST.

Совместимость с Рутокен CSP

В библиотеке rtPKCS11 реализован режим совместимости с Рутокен CSP. Совместимость между модулями заключается, в том что, ключевые пары и сертификаты, созданные в rtPKCS11, становятся доступными через интерфейс rtCSP. Так же верно и обратное: при создании контейнера и сертификата в rtCSP создаются копии объектов в rtPKCS11.

Создание объектов из rtPKCS11 в rtCSP

Совместимость достигается частичным дублированием объектов rtPKCS11 в формате rtCSP. Для корректной работы механизма совместимости должны быть выполнены следующие требования:

Требование 1

Множество возможных комбинаций значений атрибутов объектов CKO_PUBLIC_KEY, CKO_PRIVATE_KEY, CKO_CERTIFICATE (т.е. ключевой пары и сертификата) намного шире, чем множество комбинаций значений атрибутов для этих объектов в rtCSP. Поэтому не каждая ключевая пара и сертификат могут быть продублированы в rtCSP.

Для создания ключевой пары в rtCSP значения атрибутов объектов rtPKCS11 должны быть следующими.

Сочетание значений атрибутов объектов rtPKCS11, при котором возможно создание контейнера в rtCSP:

Ключевая пара в rtCSP		
	AT_KEYEXCHANGE	AT_SIGNATURE
Значение атрибутов объекта rtPKCS11: CKO_PUBLIC_KEY		
CKA_TOKEN	CK_TRUE	CK_TRUE
CKA_PRIVATE	CK_FALSE	CK_FALSE
CKA_KEY_TYPE	CKK_RSA	CKK_RSA
CKA_ENCRYPT	CK_TRUE	CK_FALSE
CKA_DERIVE	CK_TRUE	CK_FALSE
CKA_ID	< «имя контейнера» >	< «имя контейнера» >
Значение атрибутов объекта rtPKCS11: CKO_PRIVATE_KEY		
CKA_TOKEN	CK_TRUE	CK_TRUE
CKA_PRIVATE	CK_TRUE	CK_TRUE
CKA_KEY_TYPE	CKK_RSA	CKK_RSA
CKA_DECRYPT	CK_TRUE	CK_FALSE
CKA_DERIVE	CK_TRUE	CK_FALSE
CKA_ID	< «имя контейнера» >	< «имя контейнера» >
Значение атрибутов объекта rtPKCS11: CKO_CERTIFICATE		
CKA_TOKEN	CK_TRUE	CK_TRUE
CKA_PRIVATE	CK_FALSE	CK_FALSE
CKA_CERTIFICATE_TYPE	CKC_X_509	CKC_X_509
CKA_ID	< «имя контейнера» >	< «имя контейнера» >

Требование 2

Для успешного создания копий объектов в rtCSP права доступа должны быть «CKU_USER», т.о. должна быть открыта пользовательская RW сессия.

Алгоритм работы механизма совместимости из rtPKCS11 в rtCSP

После создания ключевой пары в rtPKCS11 проверяется возможность создания контейнера с ключевой парой в rtCSP. (см. Требование 1 и 2). Если выполнены требования 1 и 2, то создается контейнер с ключевой парой в rtCSP.

При создании сертификата в rtPKCS11 проверяется существование пары соответствующей данному сертификату (значение атрибута CKA_ID должно совпадать у открытого и закрытого ключей и у сертификата).

Если такая пара существует и выполняются требования 1 и 2, то проверяется существование контейнера в rtCSP для этой пары. Если контейнер создан, то в него помещается копия сертификата. Если контейнер не создан, то он создается и в него помещается копия сертификата.

При удалении сертификата в rtPKCS11, проверяется значение атрибута CKA_CAPI_ID, если оно не равно 0, то это означает, что в rtCSP существует такой сертификат. Проверяется требование 2 и сертификат в rtCSP удаляется.

При удалении открытого или закрытого ключей в rtPKCS11, проверяется требование 2 и удаляется контейнер rtCSP целиком, т.к. он не может существовать в без одного из ключей в паре.

Создание объектов из rtCSP в rtPKCS11

Совместимость достигается частичным дублированием объектов rtCSP в формате rtPKCS11.

В силу специфики rtCSP для всех контейнеров и сертификатов, созданных в rtCSP можно создать ключевые пары и сертификаты в формате rtPKCS11 (при условии достаточного места на токене).

Алгоритм работы механизма совместимости из rtCSP в rtPKCS11

После создания контейнера с ключевой парой в rtCSP создаются объекты CKO_PUBLIC_KEY и CKO_PRIVATE_KEY в rtPKCS11. (в значение атрибута CKA_ID помещается имя контейнера. В значение CKA_CAPI_ID помещается ID контейнера – для обоих ключей).

При создании сертификата в rtCSP создается копия сертификата в rtPKCS11 (в значение атрибута CKA_ID помещается имя контейнера. В значение CKA_CAPI_ID помещается ID контейнера).

При удалении сертификата в rtCSP удаляется сертификат из rtPKCS11.

При удалении контейнера в rtCSP удаляются все связанные объекты rtPKCS11 (сертификат, закрытый ключ, открытый ключ).

Разделение общих данных в механизме совместимости

В силу ограничений файловой системы Рутокен ЭЦП (нет возможности скопировать RSF-файл) и для экономии места на Рутокен - реализовано разделение общих данных двумя модулями.

Разделение общих данных на Рутокен

При работе механизма совместимости на устройствах Рутокен модули rtPKCS11 и rtCSP пользуются общими данными:

Модули rtPKCS11 и rtCSP пользуются общим файлом, в котором хранится PRIVATEKEYBLOB.

Все остальные данные необходимые для работы модулей, каждый модуль создает самостоятельно.

Разделение общих данных на Рутокен ЭЦП

При работе механизма совместимости на Rutoken ЭЦП модули rtPKCS11 и rtCSP пользуются общими данными:

RSF-файл, в котором хранится Public Key.

RSF-файл, в котором хранится Private Key.

Все остальные данные необходимые для работы модулей, каждый модуль создает самостоятельно.

Примеры использования

Исходные коды примеров использования функций библиотеки PKCS#11 с подробными комментариями входят в дистрибутив SDK.