Функции для работы с сессиями

Функции для работы с сессиями

Типичное приложение может использовать следующую очередность действий при работе с токеном (любая другая разумная последовательность действий также является допустимой):

1. Выбрать токен.
2. Сделать один или несколько вызовов функции **C_OpenSession** для получения одной или нескольких сессий с токеном соответственно.
3. Вызвать функцию **C_Login** для авторизации пользователя на токене. Так как все сессии приложения с токеном имеют общее состояние авторизации, требуется только один вызов функции **C_Login** для любой из сессий.
4. Выполнить криптографические операции, использующие сессии с токеном.
5. Вызвать функцию **C_CloseSession** для каждой из открытой приложением сессии с токеном или функцию **C_CloseAllSessions** для закрытия всех открытых сессий приложения одновременно.

Приложение может иметь одновременно несколько сессий с более чем одним токеном. Токен может иметь одновременно несколько сессий с более чем одним приложением.

Для управления сессиями стандартом предоставляются следующие функции.

C_OpenSession()

```
CK_DEFINE_FUNCTION(CK_RV, C_OpenSession)(
        CK_SLOT_ID                                  slotID,
        CK_FLAGS                                  flags,
        CK_VOID_PTR                           pApplication,
        CK_NOTIFY                                   Notify,
        CK_SESSION_HANDLE_PTR         phSession
);
```

- *slotID* - ID слота, к которому подключен токен;
- *flags* - указатель на тип сессии;
- *pApplication* - зависящий от приложения указатель на обратный вызов уведомления (в текущей реализации Рутокен параметр игнорируется);
- *Notify* - адрес функции обратного вызова уведомлений (в текущей реализации Рутокен параметр игнорируется);
- *phSession* - указатель на идентификатор (дескриптор, хэндл) новой сессии.

Назначение

Функция открывает сессию между приложением и токеном, подключенным к данному слоту.

Когда открывается сессия с использованием функции **C_OpenSession,** параметр *flags* состоит из логического ИЛИ нуля или более битовых флагов, определенных в типе **CK_SESSION_INFO.** По причинам наследования, флаг **CKF_SERIAL_SESSION** должен быть всегда выставлен, в противном случае вызов **C_OpenSession** завершится ошибкой CKR_SESSION_PARALLEL_NOT_SUPPORTED.

Ограничение количества одновременно открываемых сессий: всего, rw, read-only - зависит от токена. При попытке открыть сессию, открытие которой превысит ограничение количества одновременных сессий такого типа, вызов функции завершится с ошибкой CKR_SESSION_PARALLEL_NOT_SUPPORTED.

Если токен защищен от записи (указывается в структуре **CK_TOKEN_INFO**), то сессия может быть открыта только для чтения.

Если приложение, вызывающее **C_OpenSession**, уже имеет открытую R/W SO сессию с токеном, то попытка открытия R/O сессии завершится с ошибкой CKR_SESSION_READ_WRITE_SO_EXISTS.

Функция обратного вызова *Notify* использует библиотеку для уведомления приложения об определенных событиях. Если приложение не поддерживает обратную связь, параметр Notify должен иметь значение NULL_PTR. В текущей реализации Рутокен значение данного параметра игнорируется.

Возвращаемые значения

CKR_OK – функция выполнена успешно.

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_SESSION_COUNT,

CKR_SESSION_PARALLEL_NOT_SUPPORTED,

CKR_SESSION_READ_WRITE_SO_EXISTS,

CKR_SLOT_ID_INVALID,

CKR_TOKEN_NOT_PRESENT,

CKR_TOKEN_NOT_RECOGNIZED,

CKR_TOKEN_WRITE_PROTECTED.

Расширенные коды ошибок

Пример

Error rendering macro 'excerpt-include'

No link could be created for '3-3 Функции для работы с сессиями'.

к содержанию ↑

C_CloseSession()

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

- *hSession* - дескриптор открытой сессии.

Назначение

Функция закрывает сессию между приложением и токеном.

При закрытии сессии все созданные сессией объекты уничтожаются автоматически, даже если приложение имеет другие сессии, использующие эти объекты.

После закрытия последней сессии между приложением и токеном удачным завершением вызова функции **C_CloseSession**, состояние "залогиненности" токена для приложения вернется к публичной сессии. Любая новая сессия, открытая приложением, будет публичной сессией типа R/O или R/W.

Возвращаемое значение CKR_SESSION_CLOSED означает, пока функция выполнялась, сессия была закрыта другим вызовом **C_CloseSession**, завершившимся первым.

Возвращаемые значения

CKR_OK – функция выполнена успешно.

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_SESSION_CLOSED,

CKR_SESSION_HANDLE_INVALID.

Расширенные коды ошибок

Пример

```
CK_SLOT_ID slotID;
CK_BYTE application;
CK_NOTIFY MyNotify;
CK_SESSION_HANDLE hSession;
CK_RV rv;
.
.
application = 17;
MyNotify = &EncryptionSessionCallback;
rv = C_OpenSession(slotID, CKF_SERIAL_SESSION | CKF_RW_SESSION, (CK_VOID_PTR) &application, MyNotify,
&hSession);
if (rv == CKR_OK) {
        .
        .
        C_CloseSession(hSession);
}
```

к содержанию ↑

C_CloseAllSessions()

```
CK_DEFINE_FUNCTION(CK_RV, C_CloseAllSessions)(
        CK_SLOT_ID        slotID
);
```

- *slotID* - ID слота, к которому подключен токен.

Назначение

Функция закрывает все сессии приложения, открытые для данного токена.

При закрытии сессии все созданные сессией объекты уничтожаются автоматически.

После удачного завершения вызова функции **C_CloseAllSessions** состояние "залогиненности" токена для приложения вернется к публичной сессии. Любая новая сессия, открытая приложением, будет публичной сессией типа R/O или R/W.

Возвращаемые значения

CKR_OK – функция выполнена успешно.

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_SLOT_ID_INVALID,

CKR_TOKEN_NOT_PRESENT.

Расширенные коды ошибок

Пример

```
CK_SLOT_ID slotID;
CK_RV rv;
.
.
rv = C_CloseAllSessions(slotID);
```

к содержанию ↑

C_GetSessionInfo()

```
CK_DEFINE_FUNCTION(CK_RV, C_GetSessionInfo)(
        CK_SESSION_HANDLE hSession,
        CK_SESSION_INFO_PTR pInfo
);
```

- *hSession* - дескриптор открытой сессии;
- *pInfo* - указатель на данные сессии.

Назначение

Функция получает информацию о сессии.

Возвращаемые значения

CKR_OK – функция выполнена успешно.

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_SESSION_CLOSED,

CKR_SESSION_HANDLE_INVALID.

Расширенные коды ошибок

Пример

```
CK_SESSION_HANDLE hSession;
CK_SESSION_INFO info;
CK_RV rv;
.
.
rv = C_GetSessionInfo(hSession, &info);
if (rv == CKR_OK) {
        if (info.state == CKS_RW_USER_FUNCTIONS) {
        .
        .
        }
.
.
}
```

к содержанию ↑

C_GetOperationState()

```
CK_DEFINE_FUNCTION(CK_RV, C_GetOperationState)(
        CK_SESSION_HANDLE hSession,
        CK_BYTE_PTR pOperationState,
        CK_ULONG_PTR pulOperationStateLen
);
```

- *hSession* - дескриптор открытой сессии;
- *pOperationState* - указатель на состояние криптографических операций;
- *pulOperationStateLen* - указатель на длину значения состояния криптографических операций.

Назначение

Функция получает копию состояния криптографических операций сессии, представленную строкой байтов, для дальнейшего его восстановления функцией **C_SetOperationState.**

Если в одну сессию выполняются две криптографические операции одновременно, то состояние криптографических операций будет содержать всю необходимую операцию для восстановления обоих операций.

Попытка сохранить состояние сессии, которая на данный момент не имеет сохраняемых криптографических операций (шифрование, расшифрование, хеширование, подпись без восстановления сообщения и проверка такой подписи, а также комбинации перечисленных операций) приведет к ошибке CKR_OPERATION_NOT_INITIALIZED.

Попытка сохранить состояние сессии, в которой выполняется соответствующая криптографическая операция (или две), но которая по некоторым причинам не может быть удовлетворена (необходимые данные состояния и/или ключевая информация является неизвлекаемыми из памяти токена, например) будет завершена с ошибкой CKR_STATE_UNSAVEABLE.

Функция **C_GetOperationState** поддерживается библиотекой rtPKCS11 начиная с версии 2.30.

Возвращаемые значения

CKR_OK – функция выполнена успешно.

Стандартные коды ошибок:

CKR_ARGUMENTS_BAD,

CKR_BUFFER_TOO_SMALL,

CKR_CRYPTOKI_NOT_INITIALIZED,

CKR_DEVICE_ERROR,

CKR_DEVICE_MEMORY,

CKR_DEVICE_REMOVED,

CKR_FUNCTION_FAILED,

CKR_GENERAL_ERROR,

CKR_HOST_MEMORY,

CKR_OPERATION_NOT_INITIALIZED,

CKR_SESSION_CLOSED,

CKR_SESSION_HANDLE_INVALID,

CKR_STATE_UNSAVEABLE.

Расширенные коды ошибок

Пример

```
CK_SESSION_HANDLE hSession;
CK_MECHANISM digestMechanism;
CK_ULONG ulStateLen;
CK_BYTE data1[] = {0x01, 0x03, 0x05, 0x07};
CK_BYTE data2[] = {0x02, 0x04, 0x08};
CK_BYTE data3[] = {0x10, 0x0F, 0x0E, 0x0D, 0x0C};
CK_BYTE pDigest[20];
CK_ULONG ulDigestLen;
CK_RV rv;

.
.
/*    */
rv = C_DigestInit(hSession, &digestMechanism);
assert(rv == CKR_OK);

/*   */
rv = C_DigestUpdate(hSession, data1, sizeof(data1));
assert(rv == CKR_OK);

/*      */
rv = C_GetOperationState(hSession, NULL_PTR, &ulStateLen);
assert(rv == CKR_OK);

/*      */
pState = (CK_BYTE_PTR) malloc(ulStateLen);
rv = C_GetOperationState(hSession, pState, &ulStateLen);

/*   */
rv = C_DigestUpdate(hSession, data2, sizeof(data2));
assert(rv == CKR_OK);

/*  .     */
rv = C_SetOperationState(hSession, pState, ulStateLen, 0, 0);
assert(rv == CKR_OK);

/*      */
rv = C_DigestUpdate(hSession, data3, sizeof(data3));
assert(rv == CKR_OK);

/*   */
ulDigestLen = sizeof(pDigest);
rv = C_DigestFinal(hSession, pDigest, &ulDigestLen);
if (rv == CKR_OK) {
        /* pDigest[]   - 0x01030507100F0E0D0C */
.
.
}
```

к содержанию ↑

C_SetOperationState()

```
CK_DEFINE_FUNCTION(CK_RV, C_SetOperationState)(
        CK_SESSION_HANDLE hSession,
        CK_BYTE_PTR pOperationState,
        CK_ULONG ulOperationStateLen,
        CK_OBJECT_HANDLE hEncryptionKey,
        CK_OBJECT_HANDLE hAuthenticationKey
);
```

- *hSession* - дескриптор открытой сессии;
- *pOperationState* - указатель на сохраненное состояние криптографических операций;
- *pulOperationStateLen* - указатель на длину сохраненного состояния криптографических операций;
- *hEncryptionKey* - дескриптор ключа, используемого для текущей операции шифрования/расшифрования в восстановленной сессии (или 0, если не требуется ключ шифрования/расшифрования, или нет такой операции в восстановленной сессии, или вся необходимая ключевая информация есть в сохраненном состоянии);
- *hAuthenticationKey* - дескриптор ключа, используемого для подписи, вычисления значения MAC-кода (имитовставки), проверки подписи в восстановленной сессии (или 0, если такой ключ не требуется, или нет такой операции в восстановленной сессии, или вся необходимая ключевая информация есть в сохраненном состоянии).

Назначение

Функция восстанавливает состояние криптографических операций сессии из строки байтов, полученных функцией **C_SetOperationState.**

Состояние не обязательно должно быть получено и восстановлено в одну и ту же сессию. Однако и принимающая, и исходная сессии должны иметь общее состояние сессии (например, CKS_RW_USER_FUNCTIONS) и общий токен. Нет гарантий того, что состояние криптографических операций может быть перенесено через разные "залогиненности" или разные реализации стандарта (библиотеки).

Если функции **C_SetOperationState** предоставляется сомнительное сохраненное состояние криптографических операций, которое функция определяет как некорректное состояние (или состояние из сессии с другим состоянием сессии, или состояние с другого токена), то функция завершится с ошибкой CKR_SAVED_STATE_INVALID.

Сохраненное состояние, полученное после вызова функции **C_GetOperationState** может как содержать, так и не содержать ключевую информации для текущих криптографических операций. Если сохраненное состояние содержит операцию шифрования/расшифрования, но не содержит ключ, то он должен быть предоставлен функции аргументом *hEncryptionKey*, в противном случае **C_SetOperationState** завершится ошибкой CKR_KEY_NEEDED. Если использующийся для операции ключ сохранен в состоянии, то он может быть также передан в функцию аргументом *hEncryptionKey,* но это не обязательно.

Аналогично, если состояние содержит операцию хеширования, подписи или проверки подписи, но не содержит ключ, то он должен быть предоставлен функции в аргументе *hAuthenticationKey*, в противном случае **C_SetOperationState** завершится ошибкой CKR_KEY_NEEDED. Если использующийся для операции ключ сохранен в состоянии, то он может быть также передан в функцию через аргумент *hAuthenticationKey,* но это не обязательно.

Если функции передан неподходящий ключ (например, аргумент *hEncryptionKey* содержит ненулевой дескриптор ключа, но сохраненное состояние не содержит операций шифрования/расшифрования), то **C_SetOperationState** завершится ошибкой CKR_KEY_NOT_NEEDED.

Если функции аргументом предоставлен ключ, который **C_SetOperationState** определяется несоответствующим используемому в исходной сессии для предоставляемого состояния (ключ или хеш-значение присутствуют в сохраненном названии, например), то **C_SetOperationState** завершится ошибкой CKR_KEY_CHANGED.

Необходимость предоставления дескриптора ключа для функции **C_SetOperationState** определяется состоянием флага **CKF_RESTORE_KEY_NOT_NEEDED** в поле флагов в структуре **CK_TOKEN_INFO**. Если флаг выставлен, то для вызова функции **C_SetOperationState** никогда не потребуется дескриптор ключа, если сброшен - по крайней мере в нескольких случаях будет необходимо предоставить дескриптор ключа функции **C_SetOperationState**, следовательно, вероятно, что приложение всегда будет предоставлять подходящий дескриптор ключа при восстановлении состояния криптографических операций в сессию.

C_SetOperationState может успешно восстановить состояние в текущую сессию, даже если сессия имеет активные криптографические операции или операции поиска объектов в момент вызова **C_SetOperationState**.

Функция **C_SetOperationState** поддерживается библиотекой rtPKCS11 начиная с версии 2.30.

Возвращаемые значения

CKR_OK – функция выполнена успешно.

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_KEY_CHANGED,

CKR_KEY_NEEDED,

CKR_KEY_NOT_NEEDED,

CKR_SAVED_STATE_INVALID,

CKR_SESSION_CLOSED,

CKR_SESSION_HANDLE_INVALID.

Расширенные коды ошибок

Пример

```
CK_SESSION_HANDLE hSession;
CK_MECHANISM digestMechanism;
CK_ULONG ulStateLen;
CK_BYTE data1[] = {0x01, 0x03, 0x05, 0x07};
CK_BYTE data2[] = {0x02, 0x04, 0x08};
CK_BYTE data3[] = {0x10, 0x0F, 0x0E, 0x0D, 0x0C};
CK_BYTE pDigest[20];
CK_ULONG ulDigestLen;
CK_RV rv;

.
.
/*    */
rv = C_DigestInit(hSession, &digestMechanism);
assert(rv == CKR_OK);

/*   */
rv = C_DigestUpdate(hSession, data1, sizeof(data1));
assert(rv == CKR_OK);

/*      */
rv = C_GetOperationState(hSession, NULL_PTR, &ulStateLen);
assert(rv == CKR_OK);

/*      */
pState = (CK_BYTE_PTR) malloc(ulStateLen);
rv = C_GetOperationState(hSession, pState, &ulStateLen);

/*   */
rv = C_DigestUpdate(hSession, data2, sizeof(data2));
assert(rv == CKR_OK);

/*  .     */
rv = C_SetOperationState(hSession, pState, ulStateLen, 0, 0);
assert(rv == CKR_OK);

/*      */
rv = C_DigestUpdate(hSession, data3, sizeof(data3));
assert(rv == CKR_OK);

/*   */
ulDigestLen = sizeof(pDigest);
rv = C_DigestFinal(hSession, pDigest, &ulDigestLen);
if (rv == CKR_OK) {
        /* pDigest[]   - 0x01030507100F0E0D0C */
.
.
}
```

к содержанию ↑

C_Login()

```
CK_DEFINE_FUNCTION(CK_RV, C_Login)(
        CK_SESSION_HANDLE hSession,
        CK_USER_TYPE userType,
        CK_UTF8CHAR_PTR pPin,
        CK_ULONG ulPinLen
);
```

- *hSession* - дескриптор сессии;
- *userType* - тип пользователя;
- *pPin* - указатель на PIN-код пользователя;
- *ulPinLen* - длина значения PIN-кода пользовтеля.

Назначение

Функция авторизует пользователя на токене.

После успешной авторизации пользователя типа CKU_SO или CKU_USER все сессии приложения примут состояние R/W SO Functions или R/W User Functions или R/O User Functions. Если тип пользователя CKU_CONTEXT_SPECIFIC, то поведение функции зависит от контекста. Неправильное использование этого типа пользователя приведет к возврату значения CKR_OPERATION_NOT_INITIALIZED.

Если сессия приложения имеет активные криптографические операции или операции поиска объектов, и вызванная этим приложениям функция **C_Login** завершается успешно, то эти операции могут как остаться активными, так и нет. Поэтому перед вызовом функции любые активные операции должны быть завершены.

Если приложение, вызывающее **C_Login**, имеет открытую сессию "только для чтения" с токеном, то авторизация Администратора токена в сессии невозможна. Попытка это сделать завершится ошибкой **CKR_SESSION_READ_ONLY_EXISTS**.

Функция **C_Login** может вызываться неоднократно, без вызова функции **C_Logout** в промежутках, только (и если только) существует ключ с атрибутом **CKA_ALWAYS_AUTHENTICATE** равным CK_TRUE, и пользователю необходимо выполнить криптографические операции с его использованием.

Обработка входных параметров функции происходит следующим образом.

1. Если *ulPinLen* != 0 и *pPin* != NULL_PTR, то проверяется длина PIN-кода. Если длина не является допустимой, то возвращается значение **CKR_ARGUMENTS_BAD**. Если длина удовлетворяет условиям проверки, то проверяются права доступа для пользователя с идентификатором *userType.*
2. Если *ulPinLen* != 0 и *pPin* == NULL_PTR, то возвращается ошибка **CKR_ARGUMENTS_BAD**.
3. Если *ulPinLen* == 0 и *pPin* != NULL_PTR, то возвращается ошибка **CKR_ARGUMENTS_BAD.**
4. Если *ulPinLen* == 0 и *pPin* == NULL_PTR, то возвращается ошибка **CKR_ARGUMENTS_BAD.**

Возвращаемые значения

CKR_OK – функция выполнена успешно.

Стандартные коды ошибок:

CKR_ARGUMENTS_BAD,

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_OPERATION_NOT_INITIALIZED,

CKR_PIN_INCORRECT,

CKR_PIN_LOCKED,

CKR_SESSION_CLOSED,

CKR_SESSION_HANDLE_INVALID,

CKR_SESSION_READ_ONLY_EXISTS,

CKR_USER_ALREADY_LOGGED_IN,

CKR_USER_ANOTHER_ALREADY_LOGGED_IN,

CKR_USER_PIN_NOT_INITIALIZED,

CKR_USER_TOO_MANY_TYPES,

CKR_USER_TYPE_INVALID.

Расширенные коды ошибок

Пример

```
CK_SESSION_HANDLE hSession;
CK_UTF8CHAR userPIN[] = {"MyPIN"};
CK_RV rv;

rv = C_Login(hSession, CKU_USER, userPIN, sizeof(userPIN)-1);
if (rv == CKR_OK) {
        .
        .
        rv == C_Logout(hSession);
        if (rv == CKR_OK) {
                .
                .
        }
}
```

к содержанию ↑

C_Logout()

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

- *hSession* - дескриптор сессии.

Назначение

Функция завершает авторизацию пользователя на токене.

В зависимости от текущего типа пользователя, если вызов функции завершится успешно, каждая из сессий перейдет в состояние "R/W Public Session" или “R/O Public Session”.

После успешного завершения функции дескрипторы закрытых объектов становятся некорректными (даже если позднее пользователь зарегистрируется на токене, эти дескрипторы останутся некорректными). Кроме того, все закрытые объекты сессии, принадлежащей приложению, будут уничтожены.

Если сессия приложения имеет активные криптографические операции или операции поиска объектов, и вызванная этим приложениям функция **C_Logout** завершается успешно, то эти операции могут как остаться активными, так и нет. Поэтому перед вызовом функции любые активные операции должны быть завершены.

Возвращаемые значения

CKR_OK – функция выполнена успешно.

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_SESSION_CLOSED,

CKR_SESSION_HANDLE_INVALID,

CKR_USER_NOT_LOGGED_IN.

Расширенные коды ошибок

Пример

```
CK_SESSION_HANDLE hSession;
CK_UTF8CHAR userPIN[] = {"MyPIN"};
CK_RV rv;

rv = C_Login(hSession, CKU_USER, userPIN, sizeof(userPIN)-1);
if (rv == CKR_OK) {
        .
        .
        rv == C_Logout(hSession);
        if (rv == CKR_OK) {
                .
                .
        }
}
```

к содержанию ↑