Функции общего назначения

Функции общего назначения

C_Initialize()

```
CK_DEFINE_FUNCTION(CK_RV, C_Initialize)(
       CK_VOID_PTR       pInitArgs
 );
```

Назначение

Функция инициализирует библиотеку PKCS #11 для ее дальнейшего использования приложением. Первоначальный вызов этой функции является обязательным для работы библиотеки, иначе попытка вызова любой функции без вызова **C_Initalize** приведет к ошибке CKR_CRYPTOKI_NOT_INITIALIZED. Исключением из этого правила является вызов функции **C_GetFunctionList**, предоставляющий указатель на структуру **CK_FUNCTION_LIST**, в которой содержатся указатели на все функции, реализованные стандартом. В структуре содержатся указатели на функции, позволяющие синхронизировать работу библиотеки с помощью предоставленных пользователем механизмов, флаги, указывающие на режим синхронизации, и зарезервированный параметр.

Указатель *pInitArgs* может иметь значение NULL_PTR или указывать на структуру **CK_C_INITIALIZE_ARGS**, содержащую информацию о том, как библиотека должна себя вести при многопоточном доступе. Если приложение не будет обращаться к библиотеке несколькими потоками одновременно, то можно установить значение NULL_PTR для **C_Initialize** (последствия использования этого значения будут описаны ниже).

Если *pInitArgs* **не** NULL_PTR, **C_Initialize** должна привести его к виду **CK_C_INITIALIZE_ARGS_PTR** и затем разыменовать получившийся указатель для получения **CK_C_INITIALIZE_ARGS** полей *CreateMutex*, *DestroyMutex*, *LockMutex*,*UnlockMutex*, *flags* и *pReserved*. Для этой версии стандарта, таким образом, значение *pReserved* должно быть NULL_PTR; в противном случае **C_Initialize** вернет значение CKR_ARGUMENTS_BAD.

Если выставлен флаг **CKF_LIBRARY_CANT_CREATE_OS_THREADS**, то функция **C_Initialize** вернет ошибку CKR_NEED_TO_CREATE_THREADS, т.к. во время работы библиотеки требуется порождать новые потоки.

Стандарт PKCS#11 предполагает 4 схемы поддержки многопоточного доступа с использованием флага CKF_OS_LOCKING_OK и полей *CreateMutex*, *DestroyMutex*, *LockMutex* и *UnlockMutex:*

1. Флаг **CKF_OS_LOCKING_OK** сброшен, указатели на пользовательские функции содержат NULL. Стандарт не гарантирует корректную работу в многопоточном режиме.
   Данный режим работы в реализации для Рутокен использует внутренние механизмы синхронизации, реализованные в библиотеке (механизмы предоставлены операционной системой).
2. Флаг **CKF_OS_LOCKING_OK** выставлен, указатели на пользовательские функции содержат NULL. Стандарт предусматривает в этой ситуации использование механизмов синхронизации, реализованных в ОС для многопоточного доступа.
   Данный режим работы в реализации для Рутокен использует внутренние механизмы синхронизации, реализованные в библиотеке (механизмы предоставлены операционной системой).
3. Флаг **CKF_OS_LOCKING_OK** сброшен, указатели на пользовательские функции переданы. Стандарт предусматривает в этой ситуации использование механизмов синхронизации с помощью функций, предоставленных пользователем.
   Данный режим работы, в реализации для Рутокен, использует механизмы синхронизации с использованием функций, переданных пользователем.

4. Флаг **CKF_OS_LOCKING_OK** выставлен, указатели на пользовательские функции переданы. Стандарт предусматривает в этой ситуации использование механизмов синхронизации с помощью функций, предоставленных пользователем или механизмов синхронизации, предоставленных ОС.
5. Данный режим работы, в реализации для Рутокен, использует внутренние механизмы синхронизации, реализованные в библиотеке (механизмы предоставлены операционной системой). Функции, переданные пользователем не используются.

Вызов **C_Initialize** со значением *pInitArgs*, равным NULL_PTR, эквивалентен вызову **C_Initialize** со значением *pInitArgs*, указывающим на **CK_C_INITIALIZE_ARGS**, где значенияполей *CreateMutex, DestroyMutex, LockMutex, UnlockMutex* и *pReserved* равны NULL_PTR и поле *flags* сброшено.

Если библиотеку используют несколько приложений, то каждое из них должно вызвать функцию **C_Initialize**. Каждый вызов функции **C_Initialize** должен в итоге быть сменен одиночным вызовом функции **C_Finalize**.

Возвращаемые значения

CKR_OK – функция выполнена успешно.

Стандартные коды ошибок:

CKR_ARGUMENTS_BAD,

CKR_CANT_LOCK,

CKR_CRYPTOKI_ALREADY_INITIALIZED,

CKR_FUNCTION_FAILED,

CKR_GENERAL_ERROR,

CKR_HOST_MEMORY,

CKR_NEED_TO_CREATE_THREADS.

Расширенные коды ошибок.

Пример

Error rendering macro 'excerpt-include'

No link could be created for '2.2.4.2 Функции общего назначения'.

к содержанию ↑

C_Finalize()

```
CK_DEFINE_FUNCTION(CK_RV, C_Finalize)(
      CK_VOID_PTR
      pReserved
);
```

Назначение

Функция завершает работу с библиотекой PKCS#11. Попытка вызова любой функции после **C_Finalize** (исключение **C_Initalize** и **C_GetFunctionList**) приведет к ошибке CKR_CRYPTOKI_NOT_INITIALIZED.

Параметр *pReserved* зарезервирован под следующие версии стандарта, и в данной версии должен быть равен NULL_PTR, в противном случае функция вернет ошибку CKR_ARGUMENTS_BAD.

Если библиотеку используют несколько приложений, то каждое из них должно вызвать функцию **C_Finalize**. Вызову **C_Finalize** должен предшествовать вызов **C_Initialize**, в промежутке между вызовами этих двух функций могут выполняться другие функции, описанные в библиотеке.

Возвращаемые значения

CKR_OK – функция выполнена успешно.

Стандартные коды ошибок:

CKR_ARGUMENTS_BAD,

CKR_CRYPTOKI_NOT_INITIALIZED,

CKR_FUNCTION_FAILED,

CKR_GENERAL_ERROR,

CKR_HOST_MEMORY.

Расширенные коды ошибок.

Пример

Error rendering macro 'excerpt-include'

No link could be created for '2.2.4.2 Функции общего назначения'.

к содержанию ↑

C_GetInfo()

```
CK_DEFINE_FUNCTION(CK_RV, C_GetInfo)(
CK_INFO_PTR pInfo
);
```

Назначение

Функция позволяет получить общую информацию о библиотеке, которая возвращается в структуре типа **CK_INFO**. *pInfo* указывает на размещение этой информации.

Возвращаемые значения

CKR_OK – функция выполнена успешно.

Стандартные коды ошибок:

CKR_ARGUMENTS_BAD,

CKR_CRYPTOKI_NOT_INITIALIZED,

CKR_FUNCTION_FAILED,

CKR_GENERAL_ERROR,

CKR_HOST_MEMORY.

Расширенные коды ошибок.

Пример

```
CK_INFO info;
CK_RV rv;
CK_C_INITIALIZE_ARGS InitArgs;

InitArgs.CreateMutex = &MyCreateMutex;
InitArgs.DestroyMutex = &MyDestroyMutex;
InitArgs.LockMutex = &MyLockMutex;
InitArgs.UnlockMutex = &MyUnlockMutex;
InitArgs.flags = CKF_OS_LOCKING_OK;
InitArgs.pReserved = NULL_PTR;

rv = C_Initialize((CK_VOID_PTR)&InitArgs);
assert(rv == CKR_OK);

rv = C_GetInfo(&info);
assert(rv == CKR_OK);
if(info.version.major == 2) {
  /*      */
  .
  .
}

rv = C_Finalize(NULL_PTR);
assert(rv == CKR_OK);
```

к содержанию ↑

C_GetFunctionList()

```
CK_DEFINE_FUNCTION(CK_RV, C_GetFunctionList)(
    CK_FUNCTION_LIST_PTR_PTR
    ppFunctionList
);
```

Назначение

Функция позволяет получить список функций, поддерживаемых библиотекой. Список указателей на все поддерживаемые функции возвращается в структуре типа **CK_FUNCTION_LIST**.

C_GetFunctionList является единственной функцией стандарта PKCS#11, которая может быть вызвана до вызова функции **C_Initialize**. Это сделано для ускорения работы приложения, а также для возможности одновременного доступа приложения к нескольким библиотекам.

Возвращаемые значения

CKR_OK – функция выполнена успешно.

Стандартные коды ошибок:

CKR_ARGUMENTS_BAD,

CKR_FUNCTION_FAILED,

CKR_GENERAL_ERROR,

CKR_HOST_MEMORY.

Расширенные коды ошибок.

Пример

```
CK_FUNCTION_LIST_PTR pFunctionList;
CK_C_Initialize pC_Initialize;
CK_RV rv;

/*  C_GetFunctionList      C_Initialize */

rv = C_GetFunctionList(&pFunctionList);
assert(rv == CKR_OK);
pC_Initialize = pFunctionList -> C_Initialize;


/*   C_Initialize  */
rv = (*pC_Initialize)(NULL_PTR);
```

к содержанию ↑