

GrdLock

Функция(метод) **GrdLock** блокирует выполнение операций с ключом, давая возможность работать с ключом в монопольном режиме.

C

```
int GRD_API GrdLock(
    HANDLE hGrd,
    DWORD dwTimeoutWaitForUnlock,
    DWORD dwTimeoutAutoUnlock,
    DWORD dwMode
);
```

hGrd	хэндл, через который будет выполнена данная операция																						
dwTimeoutWaitForUnlock	таймаут, определяющий максимальный интервал времени в миллисекундах, в течении которого функция будет ожидать разблокирования ключа. Параметр используется для того, чтобы не организовывать циклы ожидания разблокирования ключа другим приложением. Если dwTimeoutWaitForUnlock равен 0xFFFFFFFF, ожидание будет происходить неограниченно долго. Если dwTimeoutWaitForUnlock равен 0, == no waiting																						
dwTimeoutAutoUnlock	параметр задает интервал времени в миллисекундах, на которое блокируется ключ. По истечении этого времени, если ключ не был разблокирован функцией GrdUnlock , драйвер снимает блокировку с ключа автоматически. Если dwTimeoutAutoUnlock равен 0xFFFFFFFF, время блокирования не ограничено и блокировка может быть снята только функцией GrdUnlock . Если dwTimeoutAutoUnlock равен 0, время блокирования устанавливается равным 10000 мс (10 секунд). Этого времени должно с запасом хватить на выполнения блока несложных операций типа - чтения данных/их модификации/запись назад. Для работы в сети данные таймауты лучше увеличивать так, чтобы за это время максимальное кол-во приложений успело выполнить данный фрагмент кода работы с ключом. Ну или писать код так, чтобы обрабатывать события автоматического разблокирования ключа.																						
dwMode	комбинация флагов GrdLM_XXXX , определяющих, доступ к каким именно операциям с ключом будет заблокирован <table><tr><td>GrdLM_Nothing</td><td>Блокируется только вызов GrdLock из другого потока, процесса или даже компьютера (при работе в сети)</td></tr><tr><td>GrdLM_Init</td><td>Блокируются операции Init</td></tr><tr><td>GrdLM_Protect</td><td>Блокируются операции Protect</td></tr><tr><td>GrdLM_Transform</td><td>Блокируются операции Transform</td></tr><tr><td>GrdLM_Read</td><td>Блокируются операции Read</td></tr><tr><td>GrdLM_Write</td><td>Блокируются операции Write</td></tr><tr><td>GrdLM_Activate</td><td>Блокируются операции Activate</td></tr><tr><td>GrdLM_Deactivate</td><td>Блокируются операции Deactivate</td></tr><tr><td>GrdLM_ReadItem</td><td>Блокируются операции ReadItem</td></tr><tr><td>GrdLM_UpdateItem</td><td>Блокируются операции UpdateItem</td></tr><tr><td>GrdLM_All</td><td>Блокируются все перечисленные операции</td></tr></table>	GrdLM_Nothing	Блокируется только вызов GrdLock из другого потока, процесса или даже компьютера (при работе в сети)	GrdLM_Init	Блокируются операции Init	GrdLM_Protect	Блокируются операции Protect	GrdLM_Transform	Блокируются операции Transform	GrdLM_Read	Блокируются операции Read	GrdLM_Write	Блокируются операции Write	GrdLM_Activate	Блокируются операции Activate	GrdLM_Deactivate	Блокируются операции Deactivate	GrdLM_ReadItem	Блокируются операции ReadItem	GrdLM_UpdateItem	Блокируются операции UpdateItem	GrdLM_All	Блокируются все перечисленные операции
GrdLM_Nothing	Блокируется только вызов GrdLock из другого потока, процесса или даже компьютера (при работе в сети)																						
GrdLM_Init	Блокируются операции Init																						
GrdLM_Protect	Блокируются операции Protect																						
GrdLM_Transform	Блокируются операции Transform																						
GrdLM_Read	Блокируются операции Read																						
GrdLM_Write	Блокируются операции Write																						
GrdLM_Activate	Блокируются операции Activate																						
GrdLM_Deactivate	Блокируются операции Deactivate																						
GrdLM_ReadItem	Блокируются операции ReadItem																						
GrdLM_UpdateItem	Блокируются операции UpdateItem																						
GrdLM_All	Блокируются все перечисленные операции																						

Набор ошибок Guardant API

Функция **GrdLock** блокирует выполнение операций с ключом для всех остальных хэндлов/приложений, работающих с этим же ключом на уровне драйвера. Функция используется в случаях, когда нужно выполнить пакет каких-либо операций, требующих синхронизации. Например обновление какой-либо ячейки памяти или для операций со счетчиками.

При выполнении блокирования ключа сначала проверяется, не заблокирован ли он из другого потока или приложения. Эта проверка также осуществляется данной функцией. В том случае, если ключ уже заблокирован, функция ожидает разблокирования ключа в течении таймаута *dwTimeoutWaitForUnlock*. Если в течении этого таймаута ключ остается заблокированным другим приложением, возвращается ошибка [GrdE_DongleBusy](#).

При блокировании операций с ключом задается время *dwTimeoutAutoUnlock*, на которое выполняется блокировка. Пои истечении этого времени драйвер автоматически разблокирует ключ. Этот таймаут задается разработчиком приложения в зависимости от потребностей. Более длинные операции, такие как считывание и запись больших объемов данных из памяти ключа требуют больше времени, следовательно и таймаут в таких случаях нужно делать больше. Также можно не ограничивать интервал автоматического разблокирования ключа (*dwTimeoutAutoUnlock = 0xFFFFFFFF*). В таком случае блокировка может быть снята только функцией [GrdUnlock](#). Это потенциально опасно тем, что при зависании приложения потребуется перезагрузка ОС для разблокирования ключа.

Операции **Login**, **Check** and **Find** заблокированы быть не могут.

На терминал-сервере работа с электронным ключом в разных сессиях идет через один драйвер ключа и соответственно блокировки должны работать так же.

Внимание!

Пара функций **GrdLock** и **GrdUnlock** не рассчитана на вызов из разных потоков. В этом случае, при вызове **GrdUnlock** из другого потока ключ не будет разблокирован (здесь можно провести аналогию с работой с критическими секциями).

C#

```
public static GrdE GrdLock(Handle grdHandle, uint timeoutAutoUnlock, uint timeoutWaitForUnlock, GrdLockMode mode)
```

grdHandle [in]

Тип: [Handle](#)

Хэндл, через который будет выполнена данная операция

timeoutAutoUnlock [in]

Тип: uint

Параметр задает интервал времени в миллисекундах, на которое блокируется ключ. По истечении этого времени, если ключ не был разблокирован функцией **GrdUnlock**, драйвер снимает блокировку с ключа автоматически.

timeoutWaitForUnlock [in]

Тип: uint

Таймаут, определяющий максимальный интервал времени в миллисекундах, в течении которого функция будет ожидать разблокирования ключа. Параметр используется для того, чтобы не организовывать циклы ожидания разблокирования ключа другим приложением.

mode [in]

Тип: [GrdLockMode](#)

Комбинация флагов GrdLockMode, которая определяет перечень заблокированных операций с ключом.

Набор ошибок Guardant API

Метод **GrdLock** блокирует выполнение операций с ключом для всех остальных хэндлов/приложений, работающих с этим же ключом на уровне драйвера. Метод используется в случаях, когда нужно выполнить пакет каких-либо операций, требующих синхронизации. Например обновление какой-либо ячейки памяти или для операций со счетчиками.

При выполнении блокирования ключа сначала проверяется, не заблокирован ли он из другого потока или приложения. Эта проверка также осуществляется данным методом. В том случае, если ключ уже заблокирован, метод ожидает разблокирования ключа в течении таймаута *timeoutWaitForUnlock*. Если в течении этого таймаута ключ остается заблокированным другим приложением, возвращается ошибка [GrdE.DongleBusy](#).

При блокировании операций с ключом задается время *timeoutAutoUnlock*, на которое выполняется блокировка. По истечении этого времени драйвер автоматически разблокирует ключ. Этот таймаут задается разработчиком приложения в зависимости от потребностей. Более длинные операции, такие как считывание и запись больших объемов данных из памяти ключа требуют больше времени, следовательно и таймаут в таких случаях нужно делать больше. Также можно не ограничивать интервал автоматического разблокирования ключа (*timeoutAutoUnlock = 0xFFFFFFFF*). В таком случае блокировка может быть снята только функцией **GrdUnlock**. Это потенциально опасно тем, что при зависании приложения потребуется перезагрузка ОС для разблокирования ключа.

Операции **Login**, **Check** and **Find** заблокированы быть не могут.

На терминал-сервере работа с электронным ключом в разных сессиях идет через один драйвер ключа и соответственно блокировки должны работать так же.

Внимание!

Пара методов GrdLock и **GrdUnlock** не рассчитана на вызов из разных потоков. В этом случае, при вызове **GrdUnlock** из другого потока ключ не будет разблокирован (здесь можно провести аналогию с работой с критическими секциями).

Java

```
public static GrdE GrdLock(Handle grdHandle, int timeoutAutoUnlock, int timeoutWaitForUnlock, GrdLockMode lockMode)
```

grdHandle [in]

Тип: [Handle](#)

Хэндл, через который будет выполнена данная операция

timeoutAutoUnlock [in]

Тип: int

Параметр задает интервал времени в миллисекундах, на которое блокируется ключ. По истечении этого времени, если ключ не был разблокирован функцией [GrdUnlock](#), драйвер снимает блокировку с ключа автоматически.

timeoutWaitForUnlock [in]

Тип: int

Таймаут, определяющий максимальный интервал времени в миллисекундах, в течении которого функция будет ожидать разблокирования ключа. Параметр используется для того, чтобы не организовывать циклы ожидания разблокирования ключа другим приложением.

mode [in]

Тип: [GrdLockMode](#)

Комбинация флагов [GrdLockMode](#), которая определяет перечень заблокированных операций с ключом.

[Набор ошибок Guardant API](#)

Метод **GrdLock** блокирует выполнение операций с ключом для всех остальных хэндлов/приложений, работающих с этим же ключом на уровне драйвера. Метод используется в случаях, когда нужно выполнить пакет каких-либо операций, требующих синхронизации. Например обновление какой-либо ячейки памяти или для операций со счетчиками.

При выполнении блокирования ключа сначала проверяется, не заблокирован ли он из другого потока или приложения. Эта проверка также осуществляется данным методом. В том случае, если ключ уже заблокирован, метод ожидает разблокирования ключа в течении таймаута *timeoutWaitForUnlock*. Если в течении этого таймаута ключ остается заблокированным другим приложением, возвращается ошибка [GrdE.DongleBusy](#).

При блокировании операций с ключом задается время *timeoutAutoUnlock*, на которое выполняется блокировка. По истечении этого времени драйвер автоматически разблокирует ключ. Этот таймаут задается разработчиком приложения в зависимости от потребностей. Более длинные операции, такие как считывание и запись больших объемов данных из памяти ключа требуют больше времени, следовательно и таймаут в таких случаях нужно делать больше. Также можно не ограничивать интервал автоматического разблокирования ключа (*timeoutAutoUnlock = 0xFFFFFFFF*). В таком случае блокировка может быть снята только функцией [GrdUnlock](#). Это потенциально опасно тем, что при зависании приложения потребуется перезагрузка ОС для разблокирования ключа.

Операции **Login**, **Check** and **Find** заблокированы быть не могут.

На терминал-сервере работа с электронным ключом в разных сессиях идет через один драйвер ключа и соответственно блокировки должны работать так же.

Внимание!

Пара методов **GrdLock** и [GrdUnlock](#) не рассчитана на вызов из разных потоков. В этом случае, при вызове [GrdUnlock](#) из другого потока ключ не будет разблокирован (здесь можно провести аналогию с работой с критическими секциями).