Kenn Nesbitt's ShareLock DLL

Calls Messages

Changes
Changes

Copyright Notice and License Agreement
Copyright Notice and License Agreement

How to Order
Ordering ShareLock

Target Platform
Any compiler/language that supports DLL's and Messaging.
There are language specific interfaces to Sharelock; if you are using

- Microsoft Visual Basic 4.0 or higher (32-bit versions only)
- Borland Delphi 2.0/3.0
- Borland C++ Builder

see the Sharelock component help file.

Installation
Installing ShareLock

Description
ShareLock is a 32-bit Windows Dynamic Link Library. The purpose of ShareLock is to prevent the use of your program after a certain number of days, a certain number of runs, or a specific date. It can also be used to simply put up a nag screen until the user registers your software.

Use
First you must pass back the handle of whatever will be processing the messages from Sharelock using the PassHandle call.

Next you use the CheckProtectionDLL call. First choose whether you wish ShareLock to work in either the specific date, number of days, number of runs, or no expire mode using the Protection parameter. If you are using the specific date mode, set the exact date your program will timeout using the Expire Date parameter. If you have selected the number of days or number of runs mode, set it using the TrialLength parameter. Each time the program is started and an unlock code has not been entered a sl_OnWithinEvalPeriod message will be generated. Use the GetTrialPeriodRemaining call to get the number of days left before the end of the trial time.

In all modes except no expire, you can specify a grace period, in days, using the GracePeriod parameter. A grace period is an extra trial time than begins after the normal trial is over. When in the grace period, the sl_OnWithinGracePeriod message will be generated.

When the trial time (and any grace period) has ended, a sl_OnExpired message will be generated.

Varied information about the protection status of your program is hidden in two seperate locations using the Windows registry. Information in both of these locations much match or else a sl_OnRegistryModified message will be fired. Select the registry locations using the RegistryLocation and RegistryLocationBackup parameters.

You can use the DoRegistration call to automate the registration process or to handle it manually, send the code to the ShareLock DLL using the InputUnlockCode call. If an incorrect unlock code is entered a sl_OnInvalidUnlockCode message is generated. The number of times the user can attempt to enter an unlock code is determined by the Tries parameter. If the Tries property is exceeded, a sl_OnExceededTries message is generated immediately following the sl_OnInvalidUnlockCode message, The current number of tries can be accessed using the GetTryNumber call. A correct unlock code will cause the sl_OnUnlocked message to be generated.

If you wish to use your own encryption engine instead of ShareLock's built in engine, use the sl_OnUserUnlockCheck message.

If using the built-in encryption make sure to set PrivateKey to a unique value.

It is possible to extend the trial length. If using the built in encryption, the length of the extension will be automatically extracted from the unlock code. If extended, the sl_OnExtended message will be generated. ShareLock will allow for a single trial length extension - if the user attemps to enter an extend unlock code more than once the sl_OnTriedToExtendAgain message will be generated.

The current date of the system is stored each time the program is run. If the user sets the system clock back to before the date the program was last run, a sl_OnClockMovedBack message is generated.

The first time your program is run, the DLL will create the required registry entries and then generate a sl_OnInitialRun message. If the DLL cannot create these entries (due to not having write permissions over a network, etc) a sl_OnCannotWriteRegistry message will be generated.

If the UseDefaultDialogs parameter is true then ShareLock will use its built in dialogs to handle all user interaction. The messages, however, will still be generated in case you wish to do further processing.

To easily display About and registration status information to the user, use the ShowAboutDialog call. You can customize the descriptions the user sees for the 'registered to' and 'registration lines' of the dialog by changing the text in the RegisteredTo, RegistrationNumber parameters. If the program is unregistered the text in the UnRegistered parameter will be displayed in place of the users name.

Note
The ShareLock DLL is started by calling the CheckProtectionDLL call each time the program is run. This call should be made before any other calls, such as GetTrialPeriodRemaining, is read.

Easiest way to use ShareLock
1. Pass the handle of whatever will process the messages that ShareLock puts out using PassHandle.
2. Call CheckProtectionDLL, telling it to use default dialogs.
3. In a routine where the user registers your software put in a call to DoRegistration.
4. In the Help-About routine of your program put in a call the ShowAboutDialog.

See the "Using ShareLock Routines - Barebones" sample app.

Sample Applications
There are example applications located in the Sample Apps subdirectory.

There are three versions apiece using ShareLock either as a component or a DLL.

- Using ShareLock routines
  - This is an example using the routines built into ShareLock while allowing you to 'peer' into the status of ShareLock. This would be used for testing different protection methods.
- Using ShareLock routines - Barebones
  - This is an example of the simplest way to use ShareLock
- Using user routines
  - This is an example using ShareLock as a protection engine while you create all the user interface.

Key Generator
If you are using the built-in encryption we have provided an unlock key generator (in Delphi and C++ Builder) located in the 'KEY GENERATOR' directory. The generator accesses a DLL (KEYGEN.DLL) which contains the encryption engine. You may write you own unlock key generator program using the DLL. Why would you want to do this? Well you might wish to extend the generator to automatically e-mail the generated keys out to customers or perhaps keep a list of all users and their keys.

The Key Generator DLL has but a single call - GenerateKey.

For more information on using the DLL see the Key Generator source code.

Contacting Nesbitt Software
You can contact Nesbitt Software on the web at www.nesbitt.com, or by E-Mail at support@nesbitt.com.

Other products by Nesbitt Software
We also sell Registration Wizard. This is a DLL/Component that automates the process of both receiving registration/credit card information from the user and then providing an unlock code to the user. All this occurs via e-mail. See our homepage for details.

Changes

Version 2.0 - July 15th, 1997
No Changes. First external release.

Version 2.01 - July 21st, 1997
The return values of sl_OnUserUnlockCheck have changed.

The return value for a full unlock is now 366 instead of 0.
The return value range for Extension Amount is now 1-365 instead of 1-255.

When it was detected that the clock had moved back, three messages were sent - 'OnClockMovedBack', 'OnSuggestTerminate' and then 'Within Trial period' (or whatever trial it was). This was wrong and potentially confusing to get the trial message after the other two. Now if the clock is moved back only the 'OnClockMovedBack' and 'OnSuggestTerminate' messages are sent.

Fixed a bug in the Extension routines. Before if your program was run after the trial period had expired and the user attempted to extend the trial, the extension was added to the original trial period, so it was possible for the program to still be expired. Now the extension time begins the day the extension is added.

Installing ShareLock

Installing ShareLock

1. Unzip the ShareLock zip file using WinZip or PKUnZip.   If you use PKUnZip, make sure you specify the -d option to preserve the directory structure.   If you use WinZip, make sure the "Use Folder Name" checkbox is checked on the Extract dialog box.

2. Copy the 'Shrlk20.DLL' file from the 'DLL' directory into your System directory (Windows 95) or System32 directory (Windows NT).

Copyright Notice and License Agreement

OR BY JURISDICTION.

LIMITATION OF LIABILITY. UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, TORT, CONTRACT, OR OTHERWISE, SHALL NSC OR ITS SUPPLIERS OR RESELLERS BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES. IN NO EVENT WILL NSC BE LIABLE FOR ANY DAMAGES IN EXCESS OFNSC'S LIST PRICE FOR A LICENSE TO THE SOFTWARE, EVEN IF NSC SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. FURTHERMORE, SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS LIMITATION AND EXCLUSION MAY NOT APPLY TO YOU.

EXPORT CONTROLS. You may not download or otherwise export or reexport the Software or any underlying information or technology except in full compliance with all United States and other applicable laws and regulations. In particular, but without limitation, none of the Software or underlying information or technology may be downloaded or otherwise exported or reexported (i) into (or to a national or resident of) Cuba, Haiti, Iraq, Libya, North Korea, Iran, Syria or any other country to which the U.S. has embargoed goods; or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Commerce Department's Table of Deny Orders. By downloading or using the Software, you are agreeing to the foregoing and you are representing and warranting that you are not located in, under the control of, or a national or resident of any such country or on any such list.

TERMINATION. Either party may terminate this Agreement immediately in the event of default by the other party. Upon any termination of this Agreement, you shall immediately discontinue the use of the Software and shall within ten (10) days return to NSC all copies of the Software and Documentation. You may also terminate this Agreement at any time by destroying the Software and Documentation and all copies thereof. Your obligations to pay accrued charges and fees shall survive any termination of this Agreement.

MISCELLANEOUS. This Agreement represents the complete and exclusive statement of the agreements concerning this license between the parties and supersedes all prior agreements and representations between them. It may be amended only by a writing executed by both parties. THE ACCEPTANCE OF ANY PURCHASE ORDER PLACED BY YOU IS EXPRESSLY MADE CONDITIONAL ON YOUR ASSENT TO THE TERMS SET FORTH HEREIN, AND NSC AGREES TO FURNISH THE SOFTWARE AND DOCUMENTATION ONLY UPON THESE TERMS AND NOT THOSE CONTAINED IN YOUR PURCHASE ORDER. If any provision of this Agreement is held to be unenforceable for any reason, such provision shall be reformed only to the extent necessary to make it enforceable, and such decision shall not affect the enforceability (i) of such provision under other circumstances or (ii) of the remaining provisions hereof under all circumstances. Headings shall not be considered in interpreting this Agreement. This Agreement shall be governed by and construed under California law as such law applies to agreements between California residents entered into and to be performed entirely within California, except as governed by Federal law. This Agreement will not be governed by the United Nations Convention of Contracts for the International Sale of Goods, the application of which is hereby expressly excluded.

U.S. Government Restricted Rights. Use, duplication or disclosure by the Government is subject to restrictions set forth in subparagraphs (a) through (d) of the Commercial Computer-Restricted Rights clause at FAR 52.227-19 when applicable, or in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, and in similar clauses in the NASA FAR Supplement.

Contractor/manufacturer is Nesbitt Software Corporation.

Contact:
Nesbitt Software Corporation
http://www.nesbitt.com/

How To Order

If you are using the evaluation version of Kenn Nesbitt's ShareLock, you must purchase a license to continue using it beyond the 30-day evaluation period.

After your order has been processed, you will receive a registered version of the software. The registered version includes the following:

- No "nag" screen
- Royalty-free license to distribute applications that include ShareLock
- Priority online technical support
- Automatic email notification of product updates
- Free minor updates
- Discounts on major upgrades

You may order Kenn Nesbitt's ShareLock from any of the following distributors:

United States
CompuServe
UK and Europe

United States

DITR Marketing, Inc.
Kenn Nesbitt's ShareLock is distributed worldwide by DITR Marketing, Inc.

DITR Marketing, Inc.
11772 Sorrento Valley Road
Suite 120
San Diego, CA 92121
USA

Voice:	(619) 259-4700
Fax:	(619) 259-5425
Email:	sales@ditr.com
Web:	http://www.ditr.com/

You may purchase ShareLock on DITR's secure web order form, or by fax, main or phone. For current version and pricing information, please visit DITR's Marketing's Software Products Page at:

http://www.ditr.com/software/

CompuServe

CompuServe

You may order Kenn Nesbitt's ShareLock online on CompuServe in the Shareware Registration forum (GO SWREG).

For current versions follow these instructions:
1. GO SWREG
2. Choose "Register Shareware"
3. Agree to the Registration Agreement
4. Choose your Geographic Region (e.g., United States)
5. Search for the Keyword "Nesbitt"

If you have any questions or difficulty using SWREG to order ShareLock, please send email to 76100,57.

UK and Europe

Grey Matter Ltd
ShareLock is distributed in the UK and Europe by Grey Matter Ltd

Grey Matter Ltd
Prigg Meadow
Ashburton
Devon
TQ13 7DF

Voice:	+44 (0)1364 654 100
Fax:	+44 (0)1364 654 200
Email:	maildesk@greymatter.co.uk
Web:	http://www.GreyMatter.co.uk

You may purchase ShareLock on Grey Matter's secure web order form, or by calling or writing. For current version and pricing information, please visit Grey Matter at:

http://www.GreyMatter.co.uk

GenerateKey call

Declaration
Delphi
```
function GenerateKey(UserName, PrivateKey: pchar; ExtensionAmount: integer):
pchar; stdcall; far; external 'KeyGen.DLL';
```

Description
The GenerateKey call call creates an unlock key. This key is only valid when using the built in encryption.

Extension amount works the following way.
For a full unlock code set ExtensionAmount to 0 (zero).
For an extension set ExtensionAmount to the length of the extension (1 - 365).

Example
```
  Code := GenerateKey('Joe Smith', 'A1B2C3', 0);
```

CheckProtectionDLL

CompanyName parameter

Declaration
Delphi
```
CompanyName: PChar;
```

```
C++
Unsigned Char * CompanyName;
```

Description
The CompanyName parameter contains the name of your company. This is only used if the UseDefaultDialogs parameter is set to true. It is used to provide contact information to the user.

Example
"My Company"
A dialog may say 'Contact   My Company for an unlock code'.

ExpireDate parameter

Declaration
Delphi
```
ExpireDate: PChar;
```

```
C++
Unsigned Char * ExpireDate;
```

Description
The ExpireDate parameter specifies the exact date that the component times out and begins to generate the sl_OnExpired message.   This only applies if the Protection parameter is set to `ptSpecificDate`. If you are not using ptSpecificDate you can leave this parameter blank. The format of the date is Day, Month, Year seperated with the '/' character. e.g. "2/15/1997".

Also see
GracePeriod parameter

GracePeriod parameter

Declaration
Delphi
```
GracePeriod: Integer;
```

```
C++
int GracePeriod;
```

Description
The value of the GracePeriod parameter determines the length (if any) of the grace period to immediately follow the end of the trial period. The grace period is usually set to a small number of days (5 or 10), and used to more forcefully ask the user to register the software. While in the grace period the sl_OnWithinGracePeriod message will be generated.

Example
During the normal trial period you might want the user to see a message like 'Please register this software - there are 8 days left in the trial'.
While in the grace period you might use a message like 'You have exceeded the trial date and are still using the software. Please register now!!!'.

Notes
This parameter has no meaning when Protection is set to ptNoExpire and may be set to 0.

ProgramName parameter

Declaration
Delphi
```
ProgramName: PChar;

C++
Unsigned Char * ProgramName;
```

Description
The ProgramName parameter contains the name of the software you are protecting.

This is used (if using the built-in dialogs) in various dialogs.

Example:
"My Program"

Protection parameter

Declaration
Delphi
`**property** Protection: integer;`

Description
The Protection property specifies the way the trial length should be determined.

The four settings (and their values) are:

```
0 - ptNumberDays
1 - ptSpecificDate
2 - ptNoExpire
3 - ptRunCount
```

To set the end of the trial period on a specific date use `ptSpecificDate`. This type of protection is usually used in Beta software when you want to be sure that a program is not used past a certain date.

To set a trial length of a specific number of days use `ptNumberDays`. If this is set and TrialLength is set to 30, then 30 days after the first time the software is run the trial will end.

To set a trial length of a specific number of program runs, use `ptRunCount`. If this is set and TrialLength is set to 30, then the software will stop working after the application has been run 30 times.

To simply have a nag screen come up every time the application is run, with no trial period, use `ptNoExpire`.

PrivateKey parameter

Declaration
Delphi
```
PrivateKey: PChar;

C++
Unsigned Char * PrivateKey;
```

Description
The PrivateKey parameter contains the unique key for your application.

If using the built in encryption engine, this is used, along with the user's name, in the generating of unlock codes. Each different program you protect should use a different private key to prevent a user using the code for one product to unlock another.

Example:
"DSF643DSF"

RegistryLocation parameter

Declaration
Delphi
```
RegistryLocation: PChar;
```

```
C++
Unsigned Char * RegistryLocation;
```

Description
The RegistryLocation parameter determines where the protection status information will be hidden in the registry. This path should be innocous sounding to keep the users from locating it too easily.

Notes:
If the information in this registry location does not match the information located at the LocationCompare parameter a sl_OnRegistryModified message will be generated to tell you that the user modified the registry setting.

Example:
```
"HKEY_CURRENT_USER\Software\SampleLocation1"
```

RegistryLocationBackup parameter

Declaration
Delphi
```
RegistryLocationBackup: PChar;
```

```
C++
Unsigned Char * RegistryLocationBackup;
```

Description
The RegistryLocationBackup parameter determines where a matching copy or the registry information pointed at by the RegistryLocation parameter is kept. This path should be innocous sounding to keep the users from locating it too easily.

Notes:
If the information in this registry location does not match the information located at the RegistryLocation property parameter a sl_OnRegistryModified message will be generated to tell you that the user modified the registry setting..

Example:
```
"HKEY_CURRENT_USER\Software\SampleLocation2"
```

TrialLength parameter

Declaration
Delphi
```
TrialLength: Integer;
```

```
C++
int TrialLength;
```

Description
The value of the TrialLength parameter determines the length in days that the program should be run for before it times out. This only applies if the Protection parameter is set to `ptNumberDays`.

Also see
GracePeriod parameter

Tries parameter

Declaration
Delphi
```
Tries: Integer;
```

```
C++
int Tries;
```

Description
The value of the Tries parameter determines how many times the user can enter a bad unlock code. Each time a bad unlock code is entered a sl_OnInvalidUnlockCode message is generated. If this happens more than Tries number of times a sl_OnExceededTries message is generated immediately following the sl_OnInvalidUnlockCode message..

Notes
Each time the program is run the user gets the Tries number of tries to enter a password.

UseDefaultDialogs parameter

Declaration
Delphi
```
UseDefaultDialogs: Integer;

C++
int UseDefaultDialogs;
```

Description
The UseDefaultDialogs parameter chooses between using the built in dialogs or letting you use your own dialogs in which routines you have trapped messages. You still must call the CheckProtectionDLL call each time the program runs.

Values
0 - False
1 - True

ShowAboutDialog

AppFilename parameter

Declaration
Delphi
```
AppFilename: PChar;

C++
Unsigned Char * AppFilename;
```

Description
The AppFilename parameter contains the full path and filename of your application, This allows ShareLock to grab the icon it uses in the upper left of the About box. If ShareLock cannot find the file it will not display an icon.

Example
"C:\APPS\MYPROGRAM.EXE"

Copyright parameter

Declaration
Delphi
```
Copyright: PChar;
```

```
C++
Unsigned Char * Copyright;
```

Description
The Copyright parameter determines what text to display below the Version line and above the "Licensed to" line. This is usually used to display a copyright message (ex: Copyright © 1997 Nesbitt Software Corp.) To include the copyright sign, ©, use the numeric keypad (press ALT + 0169), or click here to use the Windows Character Map utility.

Example
"Copyright © 1997 My Software Corp."

RegistrationNumber parameter

Declaration
Delphi
```
RegisteredNumber: PChar;

C++
Unsigned Char * RegisteredNumber;
```

Description
The RegisteredNumber parameter determines what text to display below the User Name line and above the Serial Number. This allows you to personalize the About box to your liking.

Example
"Registration Number"

RegisteredTo parameter

Declaration
Delphi
```
RegisteredTo: PChar;
```

```
C++
Unsigned Char * RegisteredTo;
```

Description
The RegisteredTo parameter determines what text to display below the Copyright line and above the User Name line. This allows you to personalize the About box to your liking.

Example
"This application is registered to"

UnRegistered parameter

Declaration
Delphi
```
UnRegistered: PChar;

C++
Unsigned Char * UnRegistered;
```

Description
The UnRegistered parameter determines what text to display on the "Registered To" line. When ShareLock has determined that the application is unregistered it will display this line.

Example
"Unregistered"

Version parameter

Declaration
Delphi
```
Version: PChar;
```

```
C++
Unsigned Char * Version;
```

Description
The Version property determines what text to display below the Product line and above the Copyright line. This is usually used to display the version number of your app.

Messages

- sl_OnCannotOpenRegistry
- sl_OnClockMovedBack
- sl_OnExceededTries
- sl_OnTrialExpired
- sl_OnExtended
- sl_OnInGracePeriod
- sl_OnInitialRun
- sl_OnInvalidUnlockCodeEntered
- sl_OnRegistered
- sl_OnRegistryModified
- sl_OnSuggestTerminate
- sl_OnTriedToExtendAgain
- sl_OnUnlocked
- sl_OnUserUnlockCheck
- sl_OnWithinEvalPeriod

sl_OnCannotOpenRegistry message

Definition
Delphi
```
sl_OnCannotOpenRegistry = WM_User + 1;
```

C++
```
#define sl_OnCannotOpenRegistry WM_User + 1;
```

Description
The OnCannotOpenRegistry message is sent if the component cannot either create the location pointed at by the RegistryLocation and RegistryLocationBackup properties, or else write protection information to registry keys under the previous properties. This is usually due to not having write permissions over a network, etc. It is up to you if you wish to treat this as a serious error or not. If ShareLock cannot create and/or write to the registry then it cannot function. You can give a message stating that your program cannot write needed information (such as window positions) to the registry and then terminate.   See the sl_OnSuggestTerminate message.

sl_OnClockMovedBack message

Definition
Delphi
```
sl_OnClockMovedBack  = WM_User + 11;
```

C++
```
#define sl_OnClockMovedBack  WM_User + 11;
```

Description
The OnClockMovedBack message is sent when the user has moved the system clock back from one running of the program to the next. Each time the program is run ShareLock stores the current date; if it detects that the current date is earlier than the recorded time this message is sent.

A common way to defeat software protection is to roll the system clock back to a time before the ExpireDate or when the TrialLength was valid. Checking for the clock being moved back can prevent this.

Note
This message is only sent if the Protection type is ptSpecificDate or ptNumberDays.

Example
In the handler for this message you should tell the user that you have detected that the clock has been moved back and then either terminate the program or let the user unlock the software. See the sl_OnSuggestTerminate message.

sl_OnExceededTries message

Definition

Delphi

```
sl_OnExceededTries = WM_User + 7;
```

C++

```
#define sl_OnExceededTries WM_User + 7;
```

Description

The OnExceededTries message is sent when, during the current running of the program, the user has entered an incorrect password more than Tries number of times.

See also

GetTryNumber parameter

sl_OnTrialExpired message

Definition
Delphi
```
sl_OnTrialExpired  = WM_User + 10;
```

C++
```
#define sl_OnTrialExpired  WM_User + 10;
```

Description
The OnTrialExpired message is sent each time the program is run and the trial time (and grace period) has run out.

sl_OnExtended message

Definition
Delphi
```
sl_OnExtended  = WM_User + 3;
```

C++
```
#define sl_OnExtended  WM_User + 3;
```

Declaration
```
SL_ONEXTENDED
hwndApp = (HWND) hWnd;              // handle to your app
wParam = (INT) length;              // length of the extension
lParam = 0;                         // not used. Must be 0
```

Parameters
hWnd
Handle to your application
wParam
The amount that the trial time has been extended.
lpsz
Not used.

Return Value
None.

Description
The OnExtended message is sent when the trial length has been extended. The amount of days/runs is passed through the ExtensionLength parameter.

```
procedure Received_OnExtended(var Msg: TMessage); message sl_OnExtended;
.
.
.
procedure TfrmTester.Received_OnExtended(var Msg: TMessage);
begin
  showmessage('The Application has been extended for ' + inttostr(Msg.wParam )
+ ' days.');
end;
```

sl_OnInitialRun message

Definition
Delphi
```
sl_OnInitialRun = WM_User + 0;
```

C++
```
#define sl_OnInitialRun WM_User + 0;
```

Description
The OnInitialRun message is sent the first time the CheckProtection call is made.

sl_OnInvalidUnlockCodeEntered message

Definition
Delphi
```
sl_OnInvalidUnlockCodeEntered  = WM_User + 6;
```

C++
```
#define sl_OnInvalidUnlockCodeEntered  WM_User + 6;
```

Description
The OnInvalidUnlockCodeEntered message is sent when an incorrect unlock code has been entered using the InputUnlockCode call.

See also
OnExceededTries

sl_OnRegistered message

Definition

Delphi

```
sl_OnRegistered  = WM_User + 12;
```

C++

```
#define sl_OnRegistered  WM_User + 12;
```

Description

The OnRegistered message is sent each time the program is run and the software has been properly registered.

sl_OnRegistryModified message

Definition
Delphi
```
sl_OnRegistryModified  = WM_User + 8;
```

C++
```
#define sl_OnRegistryModified  WM_User + 8;
```

Description
The OnRegistryModified message is sent when the registry information pointed at by the Registryocation and RegistryLocationBackup properties does not match, or else a registry key has been deleted. This is an extra form of protection to ensure that the user does not attempt to modify the protection information.

sl_OnSuggestTerminate message

Definition
Delphi
```
sl_OnSuggestTerminate  = WM_User + 14;
```

C++
```
#define sl_OnSuggestTerminate  WM_User + 14;
```

Declaration
```
SL_ONSUGGESTTERMINATE
hwndApp = (HWND) hWnd;               // handle to your app
wParam = (INT) reason;               // reason for termination
lParam = 0;                          // not used. Must be 0
```

Parameters
hWnd
Handle to your application
wParam
The reason that ShareLock suggests you terminate your application.
lpsz
Not used.

Return Value
None.

Description
The OnSuggestTerminate message is sent when ShareLock believes that the application should be shut down. By responding to this message you will be able to gracefully close down your application. wParam will be passed in with the following values:

1: Exceeded Tries - the user exceeded the alloted tries when attempting to input the registration code. See OnExceededTries message

2: CannotOpenRegistry - ShareLock could not open the registry. See OnCannotOpenRegistry message

3: Clock moved back - ShareLock has detected that the user move the system clock back. See OnClockMovedBack message

Example
Pascal
In a simple program you could respond to each reason with an Application.Terminate.

```
procedure Received_SuggestTerminate(var Msg: TMessage); message
sl_SuggestTerminate;
.
.
.
procedure TfrmTester.Received_SuggestTerminate(var Msg: TMessage);
begin
  //This is where you would gracefully exit your application
  case Msg.wParam of
    1: //Exceeded Tries
      Application.Terminate;
    2: //CannotOpenRegistry
      Application.Terminate;
    3: //Clock moved back
      Application.Terminate;
  end;
end;
```

sl_OnTriedToExtendAgain message

Definition
Delphi
```
sl_OnTriedToExtendAgain  = WM_User + 4;
```

C++
```
#define sl_OnTriedToExtendAgain  WM_User + 4;
```

Description
The OnTriedToExtendAgain message is sent when the user enters an unlock code that contains extension information, and the software has already been extended once. ShareLock will only allow a trial period to be extended once.

sl_OnUnlocked message

Definition
Delphi
```
sl_OnUnlocked  = WM_User + 5;
```

C++
```
#define sl_OnUnlocked  WM_User + 5;
```

Description
The OnUnlocked message is sent when a valid unlock code has been given to the component using the InputUnlockCode call.

Example
You could show a dialog saying something like 'Thank you for registering!'

sl_OnUserUnlockCheck message

Definition
Delphi
```
sl_OnUserUnlockCheck  = WM_User + 13;
```

C++
```
#define sl_OnUserUnlockCheck  WM_User + 13;
```

Declaration
```
SL_ONUSERUNLOCKCHECK
hwndApp = (HWND) hWnd;              // handle to your app
wParam = 0;                         // not used; must be zero
lParam = (LPARAM) (LPCTSTR) lpsz;   // address of string buffer
```

Parameters
hWnd
Handle to your application
wParam
Not used.
lpsz
Value of lParam. Points to null-terminated string that contains the user name, unlock code and company name seperated by tildes '~'.   e.g. "John Public~12345~Silly Software Inc".

Return Value

-1	No Unlock - the registration number was invalid.
1..365	Trial extended by this number of days.
366	Full Unlock

Description
The OnUserUnlockCheck message is used so that you may use your own encryption in place of ShareLock's.

To use your own encryption, create a routine that is called when this message is trapped. In this routine parse the string passed by wParam. Do your own checks to determine if the code is valid, and use the return value to tell ShareLock this result.

I suggest using the ProgramName or PrivateKey parameters in some way to generate the seed value for your engine, this way if you are protecting two seperate programs the unlock codes for each will be different.

Example
Delphi
```
procedure Received_OnUserUnlockCheck(var Msg: TMessage);  message
sl_OnUserUnlockCheck;
.
.
.
procedure TfrmTester.Received_OnUserUnlockCheck(var Msg: TMessage);
  function ParseToken(var str: string): string;
  begin
    if Pos('~', str) > 0 then
      begin
        Result := Copy(str, 1, Pos('~', str) - 1);
        str := Copy (str, Pos('~', str) + 1, 1024);
      end
    else
      begin
        Result := str;
        str := '';
      end;
  end;
var
  sStringIn: string;
  sUserName, sUserCompanyName, sUnlockCode: string;
begin
  sStringIn := StrPas(pChar(msg.lParam));
  //The code is coming in a single string seperated by '~' - let's break them
up
  sUserName := ParseToken(sStringIn);
  sUnlockCode := ParseToken(sStringIn);
  sUserCompanyName := ParseToken(sStringIn);
  //This is just an example but if the code is 'GOODCODE' then let's say it's
good.
  //If this code contained an Extension then set Msg.Result to the extension
amount ( 1 - 365 )
  if sUnlockCode = 'GOODCODE' then
    Msg.Result := 366
  else
    Msg.Result := -1;
end;
```

sl_OnWithinTrialPeriod message

Definition
Delphi
```
sl_OnWithinTrialPeriod = WM_User + 9;
```

C++
```
#define sl_OnWithinTrialPeriod WM_User + 9;
```

Description
This message is sent when the software is within the trial time.

sl_OnWithinGracePeriod message

Definition

Delphi

```
sl_OnWithinGracePeriod = WM_User + 2;
```

C++

```
#define sl_OnWithinGracePeriod WM_User + 2;
```

Description

This message is sent when the trial length has been exceeded, but the end of the grace period has not been reached.

WM_USER

{From the Win32 API Help file}

The WM_USER constant is used by applications to help define private messages.

Remarks

The WM_USER constant is used to distinguish between message values that are reserved for use by Windows and values that can be used by an application to send messages within a private window class. There are five ranges of message numbers:

Range	**Meaning**
0 through WM_USER - 1	Messages reserved for use by Windows.
WM_USER through 0x7FFF	Integer messages for use by private window classes.
0x8000 through 0xBFFF	Messages reserved for future use by Windows.
0xC000 through 0xFFFF	String messages for use by applications.
Greater than 0xFFFF	Reserved by Windows for future use.

Message numbers in the first range (0 through WM_USER - 1) are defined by Windows. Values in this range that are not explicitly defined are reserved for future use by Windows.

Message numbers in the second range (WM_USER through 0x7FFF) can be defined and used by an application to send messages within a private window class. These values cannot be used to define messages that are meaningful throughout an application, because some predefined window classes already defi ne values in this range. For example, predefined control classes such as BUTTON, EDIT, LISTBOX, and COMBOBOX may use these values. Messages in this range should not be sent to other applications unless the applications have been designed to exchange messages and to attach the same meaning to the message numbers.

Message numbers in the third range (0x8000 through 0xBFFF) are reserved for future use by Windows.
Message numbers in the fourth range (0xC000 through 0xFFFF) are defined at run time when an application calls the RegisterWindowMessage function to retrieve a message number for a string. All applications that register the same string can use the associated message number for exchanging messages . The actual message number, however, is not a constant and cannot be assumed to be the same between different Windows sessions.

Message numbers in the fifth range (greater than 0xFFFF) are reserved for future use by Windows.

DLL Calls

- CheckProtection
- DoRegistration
- GetDLLVersion
- GetExpirationDate
- GetSerialNumber
- GetStatus
- GetTrialPeriodRemaining
- GetTryNumber
- GetUserCompany
- GetUserName
- InputUnlockCode
- PassHandle
- ShowAboutDialog

CheckProtectionDLL call

Declaration
Delphi
```
procedure CheckProtectionDLL(
  Location,
  LocationCompare,
  ProgramName,
  CompanyName,
  ExpireDate:
  PrivateKey: pchar;
  TrialLength,
  GracePeriod,
  Tries,
  UseDefaultDialogs,
  Protection: integer
  ); stdcall; far; external 'Shrlk20.DLL';
```

Description
The CheckProtectionDLL call is the nexus around which ShareLock runs. This call should be made once each time the protected program is run. If Protection type is set to ptNumberRuns then each call to CheckProtection will trigger the run count to be incremented, so you should only call CheckProtection once. The CheckProtection method should normally be called during the startup phase of your program.

First choose whether you wish ShareLock to work in either the specific date, number of days, number of runs, or no expire mode using the Protection parameter. If you are using the specific date mode, set the exact date your program will timeout using the Expire Date parameter. If you have selected the number of days or number of runs mode, set it using the TrialLength parameter. Each time the program is started and an unlock code has not been entered a sl_OnWithinEvalPeriod message will be generated. Use the GetTrialPeriodRemaining call to get the number of days (or runs) left before the end of the trial time.

In all modes except no expire, you can specify a grace period, in days, using the GracePeriod parameter. A grace period is an extra trial time than begins after the normal trial is over. When in the grace period, the sl_OnWithinGracePeriod message will be generated.

When the trial time (and any grace period) has ended, a sl_OnExpired message will be generated.

Varied information about the protection status of your program is hidden in two seperate locations using the Windows registry. Information in both of these locations much match or else a sl_OnRegistryModified message will be fired. Select the registry locations using the Location and LocationCompare parameters.

When the user enters an unlock code, you send the code to the ShareLock DLL using the InputUnlockCode call. If an incorrect unlock code is entered a sl_OnInvalidUnlockCode message is generated. The number of times the user can attempt to enter an unlock code is determined by the Tries parameter. If the Tries property is exceeded, a sl_OnExceededTries message is generated immediately following the sl_OnInvalidUnlockCode message, The current number of tries can be accessed using the GetTryNumber call. A correct unlock code will cause the sl_OnUnlocked message to be generated.

The PrivateKey parameter is used (when using the built-in encryption) along with the user's name to generate an unlock key.

If you wish to use your own encryption engine instead of ShareLock's built in engine, use the sl_OnUserUnlockCheck message.

It is possible to extend the trial length. If using the built in encryption, the length of the extension will be automatically extracted from the unlock code. If extended, the sl_OnExtended message will be generated. ShareLock will allow for a single trial length extension - if the user attemps to enter an extend unlock code more than once the sl_OnTriedToExtendAgain message will be generated.

The current date of the system is stored each time the program is run. If the user sets the system clock back to before the date the program was last run, a sl_OnClockMovedBack message is generated.

The first time your program is run, the DLL will create the required registry entries and then generate a sl_OnInitialRun message. If the DLL cannot create these entries (due to not having write permissions over a network, etc) a sl_OnCannotWriteRegistry message will be generated.

If the UseDefaultDialogs parameter is true then ShareLock will use its built in dialogs to handle all user interaction. The messages, however, will still be generated in case you wish to do further processing.

To easily display About and registration status information to the user, use the ShowAboutDialog call. You can customize the descriptions the user sees for the 'registered to' and 'registration lines' of the dialog by changing the text in the RegisteredTo, RegistrationNumber parameters. If the program is unregistered the text in the UnRegistered parameter will be displayed in place of the users name.

InputUnlockCode call

Declaration
Delphi
```
procedure InputUnlockCode(UnlockCode, UserName, UserCompanyName: pchar);
stdcall; far; external 'Shrlk20.DLL';
```

Description
The InputUnlockCode call allows the software to be unlocked. sUnlockCode is a string containing the unlock code. sUserName is the users name. sUserCompanyName is the company that the user works for.

If the code is valid a sl_OnUnlocked message is generated. If the code is not valid a sl_OnInvalidUnlockCode message is generated.

ShowAboutDialog call

Declaration

Delphi

```
function ShowAboutDialog(
    Version,
    Copyright,
    RegisteredTo,
    UnRegistered,
    RegistrationNumber,
    AppFilename: pchar
    ): boolean; stdcall; far; external 'Shrlk20.DLL';
```

Description

The ShowAboutDialog method displays the About dialog.

GetTrialPeriodRemaining call

Declaration
Delphi
```
function GetTrialPeriodRemaining: integer; stdcall; far; external
'Shrlk20.DLL';
```

Description
GetTrialPeriodRemaining returns the number of days left before expiration. When GetTrialPeriodRemaining equals zero the DLL will begin to generate the sl_OnExpired message unless a GracePeriod has been set. If a GracePeriod has been specified then the call result will reset itself to show the grace period remaining and then count down the grace period left.

For Example:
Trial Period = 8
GracePeriod = 4

The first time the program is run GetTrialPeriodRemaining will equal 8.
Then 7.
Then 6.
Then 5.
Then 4.
Then 3.
Then 2.
Then 1.
Then 0.
The grace period starts and GetTrialPeriodRemaining is set to 4.
Then 3.
Then 2.
Then 1.
Then 0. Application expires.

Notes
This result has no meaning when Protection is set to ptNoExpire.

This call is only valid after the CheckProtection call has been made.

GetExpirationDate call

Declaration
Delphi
```
function GetExpirationDate: pchar; stdcall; far; external 'Shrlk20.DLL';
```

Description
GetExpirationDate returns the exact date the trial period ends. It is returned in the format Days, Months, Years seperated by "/"'s.

For Example:
"2/15/1997"

Notes
This result has no meaning when Protection is set to ptNoExpire.

This call is only valid after the CheckProtection call has been made.

GetTryNumber call

Declaration
Delphi

```
function GetTryNumber: integer; stdcall; far; external 'Shrlk20.DLL';
```

Description
This call returns the number of incorrect tries the user has made to enter the unlock code during the current running of the program.

This can be used to tell the user what try attempt they are on. i.e. 'You are on try 2 of 3'.

Note
This call is only valid after the CheckProtection call has been made.

GetDLLVersion call

Declaration
Delphi
```
function GetDLLVersion: pchar; stdcall; far; external 'Shrlk20.DLL';
```

Description
The GetDLLVersion call returns the version of the ShareLock DLL in the format MajorVersion.MinorVersion (e.g. '2.0').

GetStatus call

Declaration
Delphi
```
function GetStatus: integer; stdcall; far; external 'Shrlk20.DLL';
```

Description
The GetStatus call returns the application protection status.

0 = TrialPeriod
1 = Registered
2 = GracePeriod
3 = Expired

Note
This call is only valid after the CheckProtection call has been made.

GetUserName call

Declaration
Delphi
```
function GetUserName: pchar; stdcall; far; external 'Shrlk20.DLL';
```

Description
The GetUserName call returns the name the user entered using the InputUnlockCode method.

Note
This call is only valid after the CheckProtection call has been made.

GetUserCompany call

Declaration
Delphi
```
function GetUserCompany: pchar; stdcall; far; external 'Shrlk20.DLL';
```

Description
The GetUserCompany call returns the company name the user entered using the InputUnlockCode method.

Note
This call is only valid after the CheckProtection call has been made.

GetSerialNumber call

Declaration
Delphi
```
function GetSerialNumber: pchar; stdcall; far; external 'Shrlk20.DLL';
```

Description
The GetSerialNumber call returns the registration number the user entered using the InputUnlockCode method.

Note
This call is only valid after the CheckProtection call has been made.

PassHandle call

Declaration
Delphi
```
procedure PassHandle(AppHandle: THandle); stdcall; far; external
'Shrlk20.DLL';
```

Description
The PassHandle method allows you to pass the handle of the object that will be trapping the ShareLock messages.

DoRegistration call

Declaration
Delphi
```
procedure DoRegistration; stdcall; far; external 'Shrlk20.DLL';
```

Description
The DoRegistration call displays a dialog to the user allowing them to input their name, company name, and unlock code. This information is then automatically sent through to the registration process to determine if it is valid or not. If the sl_OnUserUnlockCheck message was defined then the user defined routine is used, else the one built into the DLL is used.