Contents

Quick Start

For a quick overview of using SLock, please see Using SLock.

About TSM Inc.

TSM Inc. is committed to promoting the spread of crypto into everyday life. Developing programs and applications (for instance, PGP) is one way to do this, but the time required to develop a polished, completed application is prohibitive and the penetration often minimal. A better way is to enable the hordes of professional and hobby developers to quickly and easily bind strong crypto into their existing applications. This series of Delphi crypto Components is therefore aimed at the professional or hobby developer with an interest in providing strong crypto in his applications.

Some applications of strong crypto are not immediately obvious, like the application of Blowfish in SLock to provide an extra level of security for the shareware author, but as the web develops more quickly and pushes ever further into our everyday lives, we will have to become used to the idea of knowledge ('a secret' like the encryption key in Slock) providing access to something.

TSM Inc. and the development of components themselves happened entirely outside of the USA, and therefore no export restrictions exist with regard to these components.

About SLock

SLock gives you, as the developer of software, the possibility to drag and drop a registration component onto a form in your application, and have the immediate ability to control the length of time your application will work. SLock allows you to limit the number of days or the number of starts that will be allowed, or set a specific date after which your application will not run. SLock also allows you to extend the trial period at any time using an unlock code. Each extension codes can only be used once, but you can give up to 99 separate extensions to an SLock component.

SLock is built upon the experience of TSM Inc. in implementing strong encryption for Delphi. Our range of encryption and security components includes 'Blowfish', ‘DES’, ‘SHA’, ‘EllipticCurve’ (public key cryptography) ‘MD5’, ‘RIPEMD-160’ and ‘RC6’. You can be secure in the knowledge that SLock is built around the tried and tested Blowfish and SHA components that offer a level of encryption which is not breakable.

SLock is designed as a full featured shareware protection system which will increase the number of registrations you receive and ensure that the users of your software will pay you for your work. We hope that you are satisfied with SLock, but in the case that you have suggestions for improvements to SLock, or find errors in the implementation, please inform us.

About the QuickLock mode

QuickLock is a simplified mode of operation in SLock that allows you to produce unlock codes for SLock applications quickly and easily. It is intended for low value, high volume applications where the loss caused by the distribution of an unlock code can be tolerated. You as a software author get the benefit of having the smallest possible maintenance overhead for the application. Higher value applications should continue using SLock in the usual mode of operation. For more details see Using SLock.

Enclosed in the SLock package you will find:

- The SLock component (‘SLock.dcu’ and ‘SLock.dcr’) and the cryptographic library (‘SlokUtil.dcu’). For Delphi4 the additional file 'SLock.bpl' is also incuded.
- An example application using SLock (‘LockTest.dpr’ in the ‘examples/LockTest’ directory)
- An example application using SLock in 'QuickLock' mode ('QuickLockTest.dpr' in the ‘examples/QuickLockTest’ directory)
- A program for creating unlock codes (‘SUnlock.dpr’ in the ‘examples/SUnlock’ directory)
- The Source code and a ready to run compiled DLL protection file, as well as various testing tools (‘dll’ in the ‘examples/dll’ directory) (See Using the DLL for more information)
- This help file (‘SLock.hlp’).

SLock features

SLock offers you the following features:

- Protection using encrypted entries anywhere in the Windows registry and in a given DLL
- Five modes of protection:
    - **Number of days** - the software will work for a given number days from the time of installation
    - **Number of unique days** - the software will work for a given number days from the time of installation. Days on which the software is not used will not be counted in the trial period
    - **Number of starts** - the software will work a given number of times
    - **Specific Expiry Date** - the software will work until a specific date is reached
    - **Time out** - the software will work for a given number of seconds each time it is started
- SLock supports Grace periods - the software will work for a given number of starts/days after it has expired, allowing you to show a ‘Register Now!’ message.
- Allows machine specific registration (software will work on only one machine)
- The Trial period can be extended up to 99 times by an extension unlock code. You can control if and when you wish to allow SLock’s trial period to be extended by. Each extension can be of up to 99 days or starts. This can be used to enable ‘software rental’ for your software.

- To register see How To Register

How To Register

SLock can be ordered from our payment processing agents, KAGI web at www.kagi.com, or by E-Mail at sales@kagi.com. TSM Inc. can also take orders direct if you wish to make your payment in US Dollars or German Marks. Further details are available on our web site www.crypto-central.com.

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 SLock
- Online technical support
- Free updates
- Discounts on major upgrades, should they become available.

Installing SLock

Delphi 2

The component can be added by simply telling deplhi where the Delphi Compiled Unit (DCU) is. You should unzip the archive you received to the final location. After doing this, select the Menu Point 'Component' and then the sub menu point 'Install component'.

The window 'Install component' will appear. Click the 'Add' button. The 'Add module' window will appear. You should click the Browse button and select the directory where you unzipped the archive, and then select the sub directory 'Delphi4'.

Important: make sure that you select the correct subdirectory for your version of Delphi. If you don't match the version of the component with the version of Delphi, you will see an error message telling you that the .pas file can not be found.

In the window 'Add module', you should set 'Files of type' to 'Delphi compiled unit (*.dcu)'. As soon as you have done this, you should be able to see the name of the component followed by the extension '.dcu' in the main browse window. Click on the .dcu file and accept it with the 'OK' button. Your selection should be taken into the 'Install component' window. Click 'OK' again in the 'Install component window'.

You should now be warned that the package will be rebuilt. Accept this. You should now see a message telling you that the component has been added.

You should now be able to find the component on the 'Crypto' Tab on the component toolbar.

Delphi 3

The component can be added by simply telling deplhi where the Delphi Compiled Unit (DCU) is. You should unzip the archive you received to the final location where you wish to have the component. After doing this, select the Menu Point 'Component' and then the sub menu point 'Install component'.

The window 'Install component' will appear. Using the Browse button next to the edit Box 'Unit File Name:' choose the directory where you unzipped the archive, and then select the sub directory 'Delphi3'.

Important: make sure that you select the correct subdirectory for your version of Delphi. If you don't match the version of the component with the version of Delphi, you will see an error message telling you that the .pas file can not be found.

In the window 'Unit File Name', you should set 'Files of type' to 'Delphi compiled unit (*.dcu)'. As soon as you have done this, you should be able to see the name of the component followed by the extension '.dcu' in the main browse window. Click on the .dcu file and accept it with the 'OK' button. Your selection should be taken into the 'Install component' window. Click 'OK' again in the 'Install component window'.

You should now be warned that the package will be rebuilt. Accept this. You should now see a message telling you that the component has been added.

You should now be able to find the component on the 'Crypto' Tab on the component toolbar.

Delphi 4

The process of installing the component uder Delphi 4 is a little different to the installation under Delphi 2 or 3.

Select the Menu item 'Component' and then the sub item 'Install packages' (**not** 'install component'). The window 'Project Options' will appear. Press the 'Add' button.

The Window 'Add Design Package' will appear. Select the directory where you installed the archive, and then select the sub directory 'Delphi4'. You should see a .bpl file listed in the main browse window. This .bpl file has the naming convention 'D4_xxxx.bpl', where xxxx is the generic TSM component identifier (e.g. 'blow' for Blowfish, 'des' for DES etc). Select the file and press 'OK'.

The 'Project Options' window will appear once again. Click 'OK'

You should now be able to find the component on the 'Crypto' Tab on the component toolbar.

NOTE: If you are installing a registered version, please be sure to remove all copies of the unregistered version before installing, as these may affect the operation of your registered software. Delphi maintains its own search path and if there are old copies on the search path, Delphi uses the first copy it finds. In most cases this will be the old copy.

Using SLock

Background

The interface to SLock is designed to be as simple to use as possible. The most important tasks in SLock are covered by two Methods which check the current status of SLock ('CheckProtection') and register SLock ('RegisterNow'). CheckProtection is the trigger for all SLock processing and must be called before attempting to read any registration information from SLock.

Other interfacing with SLock is achieved using SLock's Properties which either tell SLock how to protect your software or provide status information to the application.

The properties which provide information about how SLock should protect the application are read from the properties the first time that the application is started, and are written to the SLock's registration information. SLock's registration information is encrypted (using Blowfish in CBC mode for maximum security) and it is therefore not possible to tamper with the registry entries. If the user tries to do this, an internal checksum encrypted into the registry detects this and notifies the program. For additional security, SLock can also hold a copy of the same information encrypted into a DLL. If any of the information between the copies differs, SLock will detect this and inform the application that the registration info has been tampered with.

Feedback is provided to the application in the form of Events. You should hook the events which you want to trap in your application and provide handlers for them.

Using SLock - first steps

Before you start using SLock, you should choose whether you wish SLock to work in either the specific date, number of days, number of runs, for a timed period each time the application is started, or no expire mode. You should set the ProtectionType property depending on this decision. The way that you set the other properties will depend on which mode you decide to use.

Independent of the mode you choose to use, you should make sure that your application calls the CheckProtection method when it starts up. You can do this most easily by calling 'CheckProtection' from the 'FormCreate' event of the application's main form. This will ensure that SLock is informed each time that the application is started, and will have the chance to protect your application.

You can provide handlers for the Events which SLock can trigger, or you can inspect the value of the Status property to get the results back from SLock. As a bare minimum it is recommended that you set the OnFirstStart, OnReminder and OnExpired events. (If you are using Timout mode, you should set the OnTimedOut event).

Using SLock - first start

The first time that SLock is started, the OnFirstStart event will be triggered, regardless of the current protection mode (except 'ptNone'). This allows you to show a welcome message and give information about registering. This event will not be triggered again.

Using SLock - Expiry date mode

If you want your program to expire on a particular date, regardless of when the program is first run, you should set the ProtectionType property to 'ptExpiryDate'. The date on which SLock will expire must be entered into the ExpiryDate property. You can additionally define a grace period using the GracePeriod property. In this case the grace period defines the number of days that your program will work after the expiry date is reached.

During the trial period, SLock will trigger the OnReminder event, which allows you to optionally show a reminder message. During the grace period, SLock will trigger the OnInGracePeriod event and the and the Status will be set to 'stGrace'. When the grace period expired the OnExpired will be triggered and the status set to 'stExpired'.

Using SLock - Day count mode

If you want your program to work for a specific number of days after the first start, you should set the ProtectionType property to ‘ptDayCount’. The first time that SLock is started, the expiry date is calculated and stored in the SLock database. You should set the number of days which SLock allows using the TrialPeriod property. You can additionally define a grace period using the GracePeriod property. In this case the grace period defines the number of days that your program will work after the expiry date is reached.

During the trial period, SLock will trigger the OnReminder event, which allows you to optionally show a reminder message. During the grace period, SLock will trigger the OnInGracePeriod event and the and the Status will be set to ‘stGrace’. When the grace period expired the OnExpired will be triggered and the status set to ‘stExpired’.

Using SLock - ‘Days Used’ mode

If you want your program to work for a specific number of days after the first start *but only count the days on which the application is actually used*, you should set the ProtectionType property to ‘ptDaysUsed’. The number of days which the application can be used is stored internally as a start count that is only decremented the first time that the application is started on each unique day. You should set the number of days which SLock allows using the TrialPeriod property. You can additionally define a grace period using the GracePeriod property. In this case the grace period defines the number of days that your program will work after the expiry date is reached.

Note: This mode of operation is required by the legal definition of shareware in some countries (e.g. Germany).

During the trial period, SLock will trigger the OnReminder event, which allows you to optionally show a reminder message. During the grace period, SLock will trigger the OnInGracePeriod event and the and the Status will be set to ‘stGrace’. When the grace period expired the OnExpired will be triggered and the status set to ‘stExpired’.

Using SLock - Start count mode

If you want SLock to work for a specific number of starts after the initial start, you should set the ProtectionType property to ‘ptStartCount’. The number of starts which are allowed is set using the TrialPeriod property. You can additionally define a grace period using the GracePeriod property. In this case the grace period defines the number of times that your program will work after the expiry date is reached.

During the trial period, SLock will trigger the OnReminder event, which allows you to optionally show a reminder message. During the grace period, SLock will trigger the OnInGracePeriod event and the and the Status will be set to ‘stGrace’. When the grace period expired the OnExpired will be triggered and the status set to ‘stExpired’.

Using SLock - Timed mode

If you want to limit the amount of time that the application will run each time it is started (for example for the demo mode of a ‘SideKick’ type application), you should set the ProtectionType property to ‘ptTimed’. After you do this, you should enter the number of seconds which SLock will allow into the TrialPeriod property. The timing will start when the CheckProtection method is called.

In timed mode, there is no grace period available, and after the time period has expired, the OnTimedOut event will be triggered and the Status will be set to ‘stExpired’.

Using SLock - Checking the status of the protection

Each time the program is started, you should call the CheckProtection method. The best place to do this is in the form create method of the main form. Depending on SLock’s state, one of a number of things can happen.

The first time your program is run, the component will create the required registry entries and then generate an OnFirstStart event. If the component cannot create these entries (due to not having write permissions over a network, etc) an OnRegWriteError event 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, an OnClockMoved event is generated.

If the program is in the normal trial period, the event OnReminder will be triggered. You can define a routine in your program to hook this event to display a reminder or ‘nag’ message. You can optionally leave any event (including this) unhooked without any ill effects. You can read the number of days remaining in the trial period from the TrialPeriod property and the Expiry Date from the ExpiryDate property (in ‘Expiry Date’, ‘Day Count’ or ‘Start Count’ mode).

If the program’s normal trial period is over, but you have allowed a grace period, the OnInGracePeriod event will be triggered. You can hook this event to display a more urgent message that the user should register before the grace period ends.

If the trial period and the grace period are over, the OnExpired event will be triggered. You can hook this event to display a message telling the user how to register if they would like to use the program again.

The current status of the program can be read using the Status property. This is updated during the CheckProtection routine. The status will show if the program is in the trial period, expired, in the grace period or has been registered.

Using SLock - setting the protection options

SLock always stores its registration in a minimum of one location in the Windows registry. You can choose to also store the information in other places to make the disabling of the protection more difficult. The other locations are in a second location in the registry and in a DLL. The registration information is validated during the loading process, and if any values out of the specified range are detected, SLock will consider the registration information invalid and trigger the OnRegInfoTamper event.

If you have defined multiple copies of the information, SLock will perform the plausibility check on each of the sources, and then compare them. If any differences are found, OnRegInfoTamper will be triggered.

Name	Meaning
PoBack	also use the backup key
poDLL	also use the DLL

For more information see ProtectionOpt.

SLock can gather the information it uses to recognise whether the application has been registered or not from a number of different locations, and can be told to use any combination of these sources of information to give you the level of protection you wish. The sources of the information are set using MachineOptions.

Using SLock - setting the location of the main key

Select the registry location of the main (default) key using the RegKeyMainPath and RegKeyBackPath property. You should always do this, as SLock always requires this key.

The information will then be stored as RegKeyMainKey in the registry key defined by RegKeyMainPath.

Using SLock - setting the location of the backup key

Select the registry location of the secondary key using the RegKeyBackKey and RegKeyBackPath property. You only need to do this if you have defined ‘poBack’ in the ProtectionOpt property.

The information will then be stored as RegKeyBackKey in the registry key defined by RegKeyBackPath.

Using SLock - setting the location of the protection DLL

You can set the location of the DLL which will be used to additionally store the registration information by entering the filename or the full file path in DLLName.

- If you enter a DLL filename without a path, the DLL will be searched for in the standard directories. If more than one copy of the DLL is on the path searched, the first instance will be chosen, and any other copies will not be used. The directories searched are (in the following order):

    - in the Windows directory
    - in the Windows System directory
    - in the application's current directory.

- If you define a full path, the file will only be searched for at that one path. Note that your application can set this property during startup if the path is not known beforehand. You only need to do this if you have defined 'poDLL' in the ProtectionOpt property.

Your installation program should take care of placing the DLL in the expected directory. If you choose to define a path, you can set the location of the DLL from the program containing SLock. SLock can unfortunately not do this itself at design time, and so relies on you to do this for it. An example might be:

```
Procedure TmyForm.FormCreate (Sender: TObject);
begin
     SLock1.DLLName := ExtractFilePath(Application.Exename) + '/MyDLL.dll';
     Slock1.CheckProtection;
```

If for any reason the DLL cannot be found, SLock will report that the registration information has been tampered with and go into the expired state.

Using SLock - setting the private key

A private key should also be set so that the encryption of the various applications you produce is unique. You can enter a string into the EncryptionKey property. This private key should be between 5 and 40 characters long, and should be chosen with care as this will form both the seed for the encryption key and one of the seeds for the challenge string and the unlock code.

You need to note the encryption key down for any applications that you distribute with it because you need the encryption key to create an unlock code. This ensures that many applications can be protected by SLock - provided that the encyption key is unique for each application, a separate unlock code is needed for each.

This has an interesting side effect. If you have groups of applications, you can give them the same encryption key and key and DLL locations, causing one registration to register more than one application.

Using SLock - Registering and extending

SLock uses a challenge-response registration system. SLock gathers various information from the user's system and from the user and computes a unique string from this. The unique string (the 'challenge' string) is held in the ChallengeString property. Also contained (in the last two bytes) in the challenge string is the count of how many extensions have been made. This information is also encoded into the first eight bytes of the challenge string, so you need not be worried that the user can tamper with this.

Machine specific information can also be included into the challenge string by setting the MachineOptions property to true, meaning that SLock will only run on the single machine the unlock code was created for. You should exercise caution when using this option. If the user changes any of the machine specific information (e.g. Windows registration information) later, the program will report an error using the OnRegInfoChanged event and you will have to re-register the software for the user.

The complete the calculation of the challenge string, the registration user name should be entered into the RegName property, and the challenge string should be read and displayed. The user should quote the challenge string to you, and you can use the enclosed program 'SUnlock' to calculate an unlock code. The unlock code can either be an extension or a registration code.

Note: The challenge string also contains the RegName and this should be set before the challenge string is read. This also means that as the user is entering their user name, you have to make sure that if the challenge string is displayed in the registration window, this is always up to date.

The unlock code should be entered into SLock using the RegisterNow method. SLock will automatically detect what sort of code has been entered. If the unlock code was a valid registration code, the OnRegister event will be triggered. If the unlock code was a valid extension code, the OnExtend event will be triggered. If the code is invalid, the OnWrongUnlockCode event will be triggered.

You can easily check if the program is registered or not using the Registered property. If this is true, the software has been registered. If this is false, the program has not been registered. (You can also get this information from the Status property.

Using SLock - Registering and extending - QuickLock mode

Setting the level of protection you want is done using the MachineOptions property. For low value, high volume applications the administration overhead of providing unlock codes to users can become large, especially using such an advanced system as SLock allows. Most users are used to simply providing a name, and receiving an unlock code to match this name.

This of course means that the name/password combination can be posted in a newsgroup or warez site, but where it can be tolerated to lose a small amount of revenue to keep the admin overhead down, QuickLock mode should be used.

QuickLock mode is set by turning **all** of the MachineOptions off. This means that the Challenge String produced by SLock is depenent only on the registration name of the user and the application EncryptionKey. The Unlock code can therefore be produced without needing the Challenge String as this can be calculated by the SUnlock application.

Using SLock - Testing

Protecting your software is important - this is why you are using SLock. In order to make sure that your application is being protected properly and SLock is working the way you want it to, it is important to test the protection. Here are some Testing hints to help you in your testing.

Methods

CheckProtection

RegisterNow

GetVersion

CheckProtection

This is the main trigger for SLock, and will usually be called from the 'Form Create' Event of the main form in the application and triggers the processing of the registration information. This method must be called, or SLock will take no action.

When CheckProtection is called, the registration information is recalled from the locations defined by the ProtectionOpt property and each copy is decrypted and checked for correctness and completeness. If everything is in order, the information will then be processed and, according to the mode of operation defined in the ProtectionType property, updated. This updated information is them immediately re-crypted and wriited back to the locations.

The exact processing which is performed is driven by the property ProtectionType. For more details on using SLock and how the protection type influences the operation of SLock, please see Using SLock.

The calling convention is:

```
        SLock1.CheckProtection;
```

RegisterNow

RegisterNow passes the user entered unlock code to SLock. This unlock code can either be a registration code, in which case SLock will become registered, or an extension code which will extend the trial period. If the registration unlock code is correct, SLock will trigger the event OnRegister . If the extension unlock code is entered, SLock will trigger the event OnExtend . If the unlock code is neither a valid registration nor a valid extension code, SLock will trigger the event OnWrongUnlockCode .

The correct calling convention for ‘RegisterNow’ is:

```
        SLock1.RegisterNow(UnlockString: string);
```

Where the parameter ‘UnlockString’ is the unlock string which the user wishes to use to unlock SLock.

Note: Make sure that your application sets the user name (Using RegName) **before** reading the challenge string from ChallengeString. The reason for this is that the user name affects the value of the challenge string. See the 'Registration' window in the sample LockTest application for an example how this can be achieved.

GetVersion

GetVersion: GetVersion returns the internal version number of SLock. You will not normally need to use this method, but it can be useful when making an error report to TSM Inc. for you to also supply the version number of the SLock component you are using.

Properties

ChallengeString

DLLName

EncryptionKey

ExpiryDate

GracePeriod

MachineOptions

ProtectionType

Registered

RegKeyBackRoot

RegKeyBackKey

RegKeyBackPath

RegKeyMainRoot

RegKeyMainPath

RegKeyMainKey

RegName

Status

TrialPeriod

ChallengeString

This is the string which the user quotes to you to allow you to compute the Unlock Code. It is dependent on the following information:

- the Windows serial number (if moWinProdId is set in MachineOptions)
- the user's Registered Organisation (if moWinRegOrg is set in MachineOptions)
- the user's Registered Name (if moWinRegOwner is set in MachineOptions)
- the serial number of the C drive (if moVolSerial is set in MachineOptions)
- the volume label of the C drive (if moVolLabelis set in MachineOptions)
- the private key defined in EncryptionKey
- the registration name defined in the RegName property
- the number of extensions which have already been made

This Challenge string is then entered into the SUnlock application to create an unlock code, which should then be entered into SLock using the RegisterNow method.

DLLName

DLLName allows you to set the location of the .dll which stores registration information when the ProtectionOpt is set to include the ‘poDLL’ option. You can either specify a complete path (e.g. ‘c:\apps\testapp\lock.dll’), or just a filename (e.g. ‘lock.dll’). When the full path is specified the file will be searched for at this location. If only a filename without path is specified, the Windows directory will be searched for the file.

EncryptionKey

This is the key which will be used to base the symmetric blowfish key on. Blowfish is used to encrypt the information stored in the Windows registry. An internal algorithm is used to process this key to make it impractical to hack this key using a memory scanner.

ExpiryDate

ExpiryDate determines the expiry date of SLock when the ProtectionType property is set to 'ptExpiryDate'. When 'Protection Type' is set to any other value, this property has no effect.

GracePeriod

You can grant the user a grace period after the expiry of the program. This allows you the possibility to vary the message, or even disable certain functions in the program. The grace period is an additional number of starts, or an additional number of days after the provisional expiry. The exact meaning of this property depends on the setting of ProtectionType:

ProtectionType	**Meaning**
'ptDayCount' -	sets the number of grace days that the application can be used
'ptExpiryDate' -	sets the number of grace days that the application can be used after the expiry date
'ptStartCount' -	sets the number of grace times that the application can be used
‘ptDaysUsed’ -	sets the number of grace unique days that the application can be used
‘ptTimed’ -	no action

When the application is in the grace period, the OnInGracePeriod event is triggered when CheckProtection is called.

MachineOptions

To allow you to create an unlock code for a specific machine (rather than an unlock code for a given user) you can use any combination of various machine specific information to make the challenge string vary depending on the following hardware and software factors. As the unlock code is an response to the challenge string, you are able to produce unlock codes which will only work on a given machine:

moWinProdID: This is the Windows ‘Product ID’ (the Windows serial number), and should normally vary with every copy of Windows. This is entered during the windows set up as a ‘key’, but there are several programs available on the internet to create these keys.

moWinRegOrg: This is the company name to whom the installed Windows in registered to. This will often be left blank (for personal users).

moWinRegOwner: This is the registered owner of the installed copy of Windows. This should normally be unique for every copy of Windows.

moVolSerial: This is the volume serial number of the hard disk on which the software is currently installed. This is set ‘randomly’ when the HD is formatted, and the chances of two HDs having the same serial number is small.

moVolLabel: This is the Volume Label of the hard disk on which the current software is installed. This is set by the user during the formatting of the HD partition.

MoExtensions: Includes the number of extensions which have already been made to the application into the challenge string. You will normally only turn this off to put SLock into the QuickLock mode.

For each of these properties which is set to true, the relevant information will be included into the challenge string. For example, if you set the ‘moVolSerial’ option true, the software will be registered for one disk only. If the user later moves the data to another disk, or reformats the HD, he will have to obtain a new registration code from you

Note: Obvious though it may sound, the tighter you set the options, the more maintenance you can expect to have to perform for your customers. People upgrade their hardware (for instance their hard disks) more and more often nowadays. If you have set the machine options to include the hard disk serial number, you can expect a request for a new unlock code every now and again. You should make the decision how tight you set the protection, based on the value of the software and the attractiveness to software thieves.

QuickLock mode

QuickLock mode allows you to register an application with only the user name. This is for cheap, high volume applications where the maintenance overhead should be kept down. To do this, set all MachineOptions to 'False'. This has the effect of making the ChallengeString depend on only the RegName and the EncryptionKey.

ProtectionOpt

ProtectionOpt specifies what information will be retrieved to check the registration status of the application. The Main copy of the registration information in the Windows registry (specified by RegKeyMainPath and RegKeyMainKey) is always included, and forms the basis of the information.

Additionally you can choose to have a backup copy also stored in the Windows registry by specifying the option ‘poBack’. If this is set, a copy of the information will be read from the location in the registry specified by RegKeyBackPath and RegKeyBackKey.

If you have chosen to also include to store information in the DLL, you should make sure that you also set the property DLLName.

ProtectionType

ProtectionType determines which of the protection schemes will be used. Possible values are:

```
type TProtectionType = (ptNone, ptExpiryDate, ptStartCount, ptDayCount, ptDaysUsed,
ptTimed);
```

ProtectionType	Meaning
'ptNone'	no protection
'ptDayCount'	sets the number of days that the application can be used
'ptExpiryDate'	sets a fixed date after which the application will expire
'ptStartCount'	sets the number of times that the application can be used
‘ptDaysUsed’	sets the number of unique days that the application can be used
‘ptTimed’	each time SLock starts, it will allow a certain number of seconds before expiring

For a more complete introduction to using SLock, see Using SLock.

Registered

Registered shows if SLock has been given a valid Unlock Code. Is SLock has been correctly registered, this property will be true after CheckProtection has been called, otherwise it is false.

For more detailed information what the registration status is, you can also use Status.

RegKeyBackRoot

RegKeyBackRoot allows you to define under which registry root key the backup copy of the registration information should be stored. For most purposes the default option should be OK, but should you want to place the key somwhere else, you can do so using this option.

Valid root keys are:

- HKEY_CURRENT_USER
- HKEY_CLASSES_ROOT
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_CONFIG
- HKEY_DYN_DATA

The predefined items of this property are those shown above, without the leading 'HKEY_'.

The key path and the key name are defined by RegKeyBackPath and RegKeyBackKey respectively. A similar option also exists for the main key RegKeyMainRoot.

RegKeyMainRoot

RegKeyMainRoot allows you to define under which registry root key the main copy of the registration information should be stored. For most purposes the default option should be OK, but should you want to place the key somwhere else, you can do so using this option.

Valid root keys are:

- HKEY_CURRENT_USER
- HKEY_CLASSES_ROOT
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_CONFIG
- HKEY_DYN_DATA

The predefined items of this property are those shown above, without the leading 'HKEY_'.

The key path and the key name are defined by RegKeyMainPath and RegKeyMainKey respectively. A similar option also exists for the backup key RegKeyBackRoot.

RegKeyBackPath

The backup path under which the encrypted registration information will be stored in the windows registry. Two copies are kept (to help detect tampering) and these should be held in different locations. The main copy is RegKeyMainPath.

The key name itself is defined by the property RegKeyBackKey. The combination of these two properties allows full flexibility in the way that information is stored in the Windows registry.

RegKeyMainPath

The main path under which the encrypted registration information will be stored in the Windows registry. Two copies are kept (to help detect tampering) and these should be held in different locations. The main copy is RegKeyBackPath.

The key name itself is defined by the property RegKeyMainKey. The combination of these two properties allows full flexibility in the way that information is stored in the Windows registry.

RegKeyBackKey

The name of the key which will be stored by SLock. This is stored at the path defined by RegKeyBackPath.

A second copy (to help prevent tampering) is stored at: RegKeyMainPath\RegKeyMainKey.

RegKeyMainKey

The name of the key which will be stored by SLock. This is stored at the path defined by RegKeyMainPath.

A second copy (to help prevent tampering) is stored at: RegKeyBackPath\RegKeyBackKey.

RegName

The User Name should be entered into this property to allow SLock to compute the ChallengeString. As soon as the user name is entered, the new challenge string will automatically be calculated.

After a call to CheckProtection this property can be read to retrieve the name of the registered owner of the software if the Status is ‘stRegistered’.

Status

Shows the current status of SLock, and is set during the processing of the CheckProtection event. Possible values are:

```
type TStatus = (stNotExpired, stGrace, stExpired, stRegistered);
```

ProtectionType	**Meaning**
‘stNotExpired'	the application is in the normal unregistered state.
'stGrace'	the application is in the grace period
'stExpired'	the application has expired
'stRegistered'	the application has been registered

For a quick check if SLock has been registered or not, use the Registered property.

TrialPeriod

TrialPeriod determines the number of days or starts that SLock will allow the application to run before it expires. The exact meaning of this property depends on the ProtectionType which you have selected.

ProtectionType	**Meaning**
'ptDayCount' -	sets the number of days that the application can be used
'ptExpiryDate' -	no action
'ptStartCount' -	sets the number of times that the application can be used
‘ptDaysUsed’ -	sets the number of unique days that the application can be used
‘ptTimed’ -	no action

After the application expires, the GracePeriod property determines the length of the grade extension period.

When the application is in the trial period, the OnReminder event is triggered when CheckProtection is called.

Events

OnClockMoved

OnExpired

OnExtend

OnFirstStart

OnInGracePeriod

OnRegInfoChanged

OnRegInfoTamper

OnRegister

OnRegWriteError

OnReminder

OnTimedOut

OnWrongUnlockCode

OnClockMoved

OnClockMoved is triggered when CheckProtection is called if the system clock has been tampered with, for instance if the clock is set back past the creation date of SLock or the system clock has been moved back since the last start.

OnExpired

OnExpired is triggered when CheckProtection is called if the component has expired. You can chose to use this event to display a message telling the user how to register and then exit the program.

OnExtend

OnExtend is triggered when RegisterNow is called and a valid extension code has been entered into SLock.

The Status of SLock will be set to ‘stGrace’ and SLock will consider itself to be in the grace period next time it starts, regardless if there was any unexpired time in the normal trial period or not.

OnFirstStart

OnFirstStart is triggered when CheckProtection is called the fist time the component is used. You can use this to display a welcome message or ‘readme’ file. This event will only be triggered the first time tha t your application is run on a new machine, and will not be triggered again.

OnInGracePeriod

OnInGracePeriod is *t*riggered when CheckProtection is called and the component has expired, but is still in the grace period (if one is defined).

OnRegInfoChanged

OnRegInfoChanged is triggered CheckProtection   is called and the user information or the machine information is changed (only when the MachineOptions property is set true). You can chose to display a warning message and possibly exit the program.

OnRegister

OnRegister is triggered when RegisterNow is called and a correct registration unlock code is entered into SLock. You can use this to display a 'thank you' message.

OnRegInfoTamper

OnRegInfoTamper is triggered when CheckProtection is called if the registry is tampered with or damaged. You can chose to display a warning message and optionally end the program.

OnRegWriteError

OnRegWriteError is triggered when CheckProtection   is called if there was a problem writing to the Registry. SLock uses a read after write validation to ensure that the information is correctly stored in the Windows registry.

OnReminder

OnReminder is triggered each time that the CheckProtection method is called while SLock has not been given a valid unlock code. You can use this to display a nag message.

OnTimedOut

OnTimedOut is triggered the number of seconds defined in TrialPeriod after the CheckProtection method is called. The Status property is set to ‘stExpired’ before this event is called.

OnWrongUnlockCode

OnWrongUnlockCode is triggered when the RegisterNow method is called with an invalid unlock code. You can use this to display a warning message and optionally exit the program.

Testing hints

Just as you would never ship an application before testing it, you should test how your application interacts with SLock.

There are three important things to check:

- Make sure that the settings on the object inspector are correct.
- Check to see that the trial length and type of protection are set.
- Make sure that you have specified valid registry locations.

See the properties RegKeyMainKey and RegKeyMainPath for more information of what to keep in mind when choosing a registry location.

The most important thing: Test your application just as if you were a user.

If you have set SLock to expire after a number of days then bring up the system clock and advance it a few days to verify that the trial period has decreased that amount. Switch the clock to a period after the expiration date and verify that SLock has expired (or the grace period has started). Now try to register your application using invalid codes and make sure that your application shuts down so that it cannot be used after it has been expired. Try registering your application with known good codes to make sure it unlocks properly.

If you have set SLock to expire after a number of runs then close down and restart your application and verify that the trial period decreases and finally expires.

To reset the 'state' of the protection you will need to clear out the registry keys that are being used. You can use the Windows RegEdit program. To use RegEdit, select the Windows 'Start' button, then 'Run' and type 'RegEdit'.

SLock will do an excellent job protecting your application but only if it is set up properly. The only way to make sure of this is to test your application before shipping it.

Using the DLL

Full source code to the DLL is included in the Slock package. This allows you to make use of the DLL for your own purposes should you wish to. In any case, it is wise to compile the DLL for your version of Delphi to ensure compatibility.

The DLL has only two functions in its basic state:

```
function GetRegInfo: TBuffer; stdcall;
function GetBuffAddr(dllName: Pchar): integer; stdcall;
```

GetRegInfo returns an array of 256 Bytes which will be filled with the registration information by SLock if you set ProtectionOpt to include 'poDLL'.

GetBuffAddr returns the address (the offset from the start of the DLL) of the buffer in the DLL, and which allows Slock to write into the DLL. Because SLock needs to be able to write access to the DLL, it cannot be statically linked, and must therefore use the slightly more complicated dynamic linking.

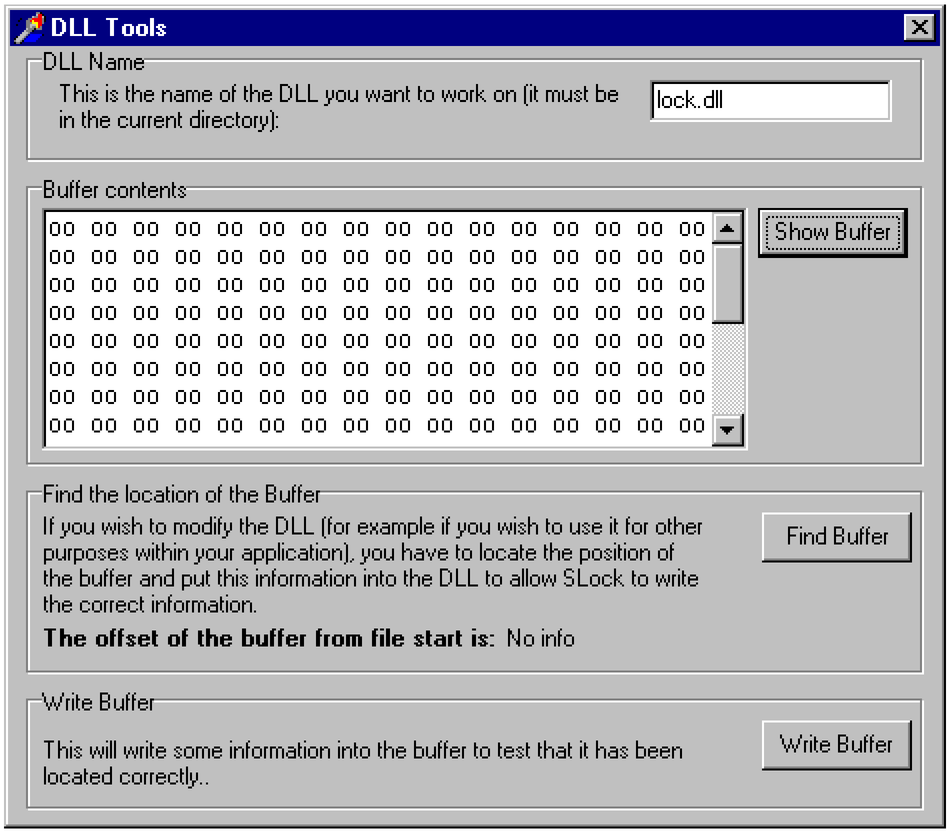
If you wish to add functionality into the DLL, for example if you wanted to use it to hold other routines apart from those supplied in the basic package, you should remember two things:

- Due to the nature of Windows, it is difficult to quickly find the modified position of the buffer in the DLL. SLock does a simple search through the DLL looking for a header ('Buf:') and knows that the next byte after the colon is the start of the buffer. If you add functionality to the DLL, the position of the buffer will change.

  To make the search for the buffer as quick as possible, you should update the offset in the GetBuffAddr routine so that the search is started further through the file. For the maximum possible speed, you could simply hard-code the value into the function.
- **Note:** The DLLTest project provides tools for finding the position of the buffer in the DLL.
- Don't statically link the DLL!

DLLTest - testing program for the DLL

The main window of the testing program for the DLL is shown below. The main features are:



This is the name of the DLL you want to work on (it must be in the current directory): lock.dll

```
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
```

If you wish to modify the DLL (for example if you wish to use it for other purposes within your application), you have to locate the position of the buffer and put this information into the DLL to allow SLock to write the correct information.

The offset of the buffer from file start is: No info

This will write some information into the buffer to test that it has been located correctly..

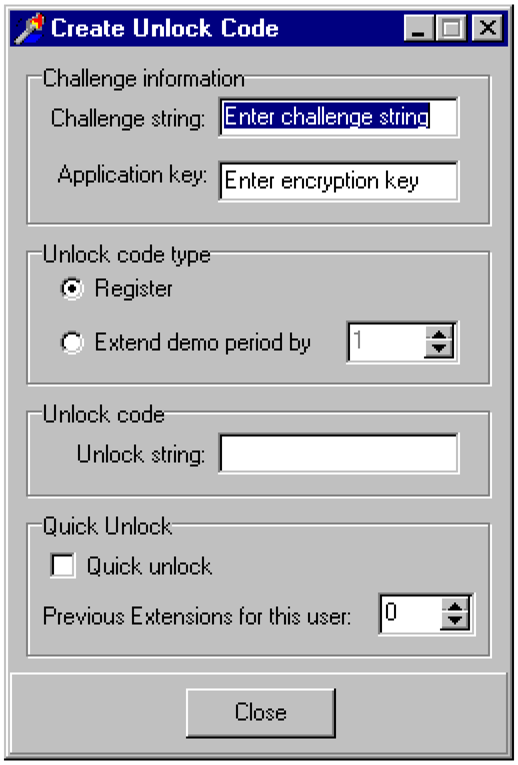
- You can specify the name of the DLL, which you want to test in the 'DLL Name' edit box
- You can show the contents of the buffer in the DLL using the 'Show Buffer' button
- You can find the start position of the Buffer in the DLL (the offset from the start of the file) by using the 'Find Buffer' button - this can take a few seconds, depending on the size of the DLL
- You can test that you are able to write to the buffer in the DLL using the Write Buffer button.

The buffer is encrypted in normal use, so although you will be able to read the contents of the buffer using the DLLtest program, you will not be able to interpret it, even though you know the EncryptionKey. SLock performs an amount of internal processing on the encryption key before it is used.

Note: As a normal user of SLock, you will usually not have to get involved with the internal workings of the DLL and the buffer. This information is therefore provided for expert users, and should not be used if you are unsure of the consequences of recompiling the DLL.

SUnlock

The SUnlock application allows you to quickly and easily create unlock codes for the applications you produce using SLock. Full source code is included in the application, so you are able to modify SUnlock to meet your own specific needs, (for instance if you want to create a drop down list for the application keys).



Challenge string: Enter challenge string

Application key: Enter encryption key

(● Register) ○ Extend demo period by 1

☐ Quick unlock

Previous Extensions for this user: 0

Normal mode

In the normal mode of operation, you will create an unlock code by entering the ChallengeString from the purchaser of your software into the 'Challenge string' edit box. You should also enter the EncryptionKey which you have created for the application.

After doing this, you should decide whether you wish to register the application or just provide an extension to it. In the normal case you will leave the 'Register' radio button checked to make an unlock code. After setting these options, you can read the unlock code from the 'Unlock code' edit box.

QuickLock mode

To produce a QuickLock unlock code, you should check the 'Quick Unlock' check box. This will change the 'Challenge string' caption into 'User name', and will disable the 'Unlock code type' group.

In this mode you will enter the name of the user and the EncryptionKey. The unlock code will be shown in the usual place.

The QuickUnlock mode also allows you to produce extensions. If you want to create a further extension for a user (this will not usually happen) then you will have to track the number of extensions the user has had, and when producing an unlock code in QuickLock mode you will have to enter the number of previous extensions.

Version history

V1.00 (14.5.98)

- Original component

V1.00a (16.5.98)

- Bugfix - the application private key was added into the challenge string and the unlock string to ensure that more than one application can not be unlocked with the same unlock code.

V1.01 (21.5.98)

- Additional machine specific options were built in on the request of several users (Disk ID, Disk Serial number).

V1.01a (28.5.98)

- ‘Utils’ was renamed to ‘SlokUtil’
- Timed mode of operation was added.

V1.02 (12.7.98)

- The storage locations of the keys in the Windows Registry was made more flexible by adding variable names for the keys. (They were fixed before.) The additional properties were added to allow these to be set. (RegKeyMainKey and RegKeyBackKey). To avoid confusion the name of the properties which define the paths of the keys was changed. (RegKeyMainPath and RegKeyBackPath).

V1.10 (Point release 15.8.98)

- Delphi4 support added
- The Property 'Status' is now set before calling the appropriate event. The order was originally to call the event and then set the property, which makes no sense, as sometimes the property is used in the event.
- In addition to the existing storage locations of the registration information, a third location has been added to make it very much more difficult to bypass SLock security. A third copy of the registration information can now be stored in a DLL file which can either be located in the Windows directory, or at another given location. The property DLLName allows you to specify the location of the DLL. If this is given without a path, the DLL will be searched for in the Windows directory.
- An additional property has been added to allow more flexibility in using the three locations of registration information. 'ProtectionOpt' allows you to set if the Backup registry location and or the DLL should be used to store the registration information. The Main registry location is always used.
- In certain countries (e.g. Germany) the definition of Shareware is that the program should be used for a number of distinctive days - any day on which the software is not used should not be taken from the trial period. SLock was not able to support this mode of operation, but has now been extended to have an additional protection mode 'ptDaysUsed'. In this mode SLock can be used for the given number of days, and any day on which SLock is not used will not be counted in this period.
- The name of the Event 'RegistryTamper' has been changed to 'RegInfoTamper'. The properties for determining the trial period have been rationalised and renamed. There were a number of different properties to determine the period ('AllowedDays', 'AllowedStarts' etc.) These have now been reduced to the single property 'TrialPeriod'. This should make SLock a little easier to understand.

V1.10a (22.8.98)

- QuickLock mode added

V1.11 (Point release 25.9.98)

- The option to change the roots of the registry information added 'RegKeyMainRoot' and 'RegKeyBackRoot'.
- Minor bug fixes