Delphi A-Z (as main source)

# Proteus

Do you have your own commercial app that you are selling, or do you plan to do that? If yes, read on, if not, skip the whole chapter.

### What is Proteus?

Proteus is a licensing system that I designed. It helps us to convert a dull Delphi program to a commercial program that we can deliver as a trial/shareware.

Multiple usage scenarios are possible:

* Trial license  
  The customer receives a trial version of your software. The trial could be fully functional or “crippled” - for example, it will not save the data/project to disk. Upon purchase, we only need to email a key (text) to the customer to upgrade his license from Trial to a Rent or Permanent license.
* Rent license  
  The customer can rent a license (purchases a license that works for only x days). Later, the customer can also upgrade to a permanent license.
* Permanent license

A Permanent license never expires.

* Modules

You can send to the customers a license key that unlocks all or only some of parts/modules of your program.

## What can Proteus do for us?

### Automates the process of selling of your app

If we use an external payment processor, Proteus empowers us to fully automate the process of selling our programs.

For example, I use BlueSnap.com as a payment processor. For products where I have to generate keys each day, I pre-generate 5000 unlock keys in my Proteus Key Generator app I and upload them on BlueSnap.

BlueSnap and Proteus take now care of the rest. Upon purchase, each customer automatically receives from BlueSnap one of these unlock keys. The customer enters the key into the program, where Proteus validates the key and unlocks the program.

So, I fully automated the whole process. I don’t even know who my customers are.

When the keys are all gone, I get a notification from BlueSnap and I generate a new batch and upload them.

### Prevents fraud

We can take measures if a license key is leaked by a customer on serial/cracking websites, by immediately invalidating that key. In case of a “stolen” license key, Proteus can lockdown the program. For this, see the OnKeyStolen event.

Also, measures have been put in place to make sure the customer does not try to temper with the license key it received or with the installation date (for Trial versions).

### Proteus can even do marketing

Proteus helps at marketing also. With proteus we can target a group of people (Paying customers/Trial users/Demo users/All).

For example, we could show news and updates to already paying customers while showing discounts to people that haven’t purchased yet a license.

## How to use it?

### Typical usage scenario

Drag and drop Proteus control onto your form. Deliver the application as trial to your customers where it will work for 30 days (Trial period) then it will switch to Demo mode.

In the Demo mode, we could cripple program's functionality severely, or show only a "nag" screen to encourage the customer to purchase a license.

Once the customer pays for a license, we can send them a key to unlock the program. This key will switch the program from Demo mode to Purchased (fully functional) mode.

### Is it difficult to integrate Proteus into my app?

We will not see much code in this chapter about Proteus because we only must write a single line of code to integrate Proteus into our applications. Simply call “TProteus.Initialize” at application start-up.

Optionally, we can do some customizations by writing code for the event handlers for TProteus.OnSwitchedToDemo and TProteus.OnLicenseFallBack. Well, not even that. For lazy people, Proteus already offers predefined messages for those event handlers, such as

* ShowKeyAccepted,
* ShowKeyUnknown,
* ShowSwitchToDemo,
* SwitchToDemo,
* SwitchToDemoFallBack

### How to convert a regular Delphi app to a Trialware in 5 minutes?

1. Install the Proteus package. Drop a TProteus control on your form. Call “TProteus.Initialize” during application initialization.
2. In your program’s initialization routine, check the status of the currently installed certificate (TProteus.CurCertif). If the certificate is in:

* Full mode: Write code to let the user use your program
* Demo mode: Write code to prevent the user from using your program (and encourage the user to purchase a license)

1. Compile and deliver your application.

Note: The user will receive a notification from Proteus when the trial period expires. Write an event handler for that and decide what to do. For example, we could announce to the user that the trial has expired, we could redirect it to the Purchase webpage, we could shut down the program, we could cripple the program (allow all function except “Save work”). You are limited only by your imagination.

Full blown demo app also available.

## Proteus User Manual

### DEFINITIONS

#### Certificate

A certificate is a record that informs your program about the status of the license. It contains info such as:

* Certificate expiration date
* Certificate state: trial/demo/full
* Modules unlocked (available for use)
* Customer related data (customer name, organization)
* etc

The certificate can be:

* Permanent (it unlocks the program forever)
* Temporary (it unlocks the program for a predetermined period)

A program can have multiple certificates installed in its storage area, but only one is active. This one is called the *current certificate*. The user cannot/should not switch between certificates. Instead, Proteus itself implements the logic to choose the appropriate certificate.

For example, an application was delivered with a default “Trial certificate”. Upon purchase of a license, a key is sent to the user and the user enter the key into the program. Therefore, a new certificate (“Full”) appears in the system. Proteus will choose the better (the “Full”) certificate. The lesser certificate (the “Trial”) will be ignored until the current certificate expires (if ever).

Also, if the user was running on a temporary license like “Rent 30 days”, after 30 days, upon certificate expiration the program will switch from this certificate to another certificate IF available. For example, if the user still has time left on his Trial certificate, Proteus will switch to that certificate and run until whatever time was left, is used up. If the Trial certificate has no valid time left on it, the program will switch to Demo mode.

This allows incredible flexibility.

#### Flexible updates

The user can download a new version (update) from your website when available.

If the customer already had a valid certificate installed on his computer (Full or Rent), upon installing the update, the new downloaded version will automatically use the better certificate.

#### Unlock key

The unlock key (or simply “the key”), is a text representation of a certificate (see “certificate” definition above). Proteus generates it by passing certificate’s binary data through a MIME encoder to get a human-readable string. Don’t worry, the binary data is first encrypted, and a checksum is inserted. So, the certificate cannot be easily tempered with. Proteus allows you also to call your own encryption algorithms if you want.

Upon purchasing a license, the user receives a key from us. The user enters the key into the program. You can use the existing 'EnterKey' method or define your own GUI.

If Proteus validates the key ok, the program is unlocked. If not, a friendly error msg is returned to the user.

Simply put, the certificate and the unlock key are one and the same thing. The small difference is that the certificate is a binary Delphi structure while the key is a human readable string (so it can be sent to the customers by email). We will use the two terms interchangeably in this document.

You can also view the text key as a “recipient” or “transporter” of the binary certificate.

We can choose if the key will unlock only the current version or also other versions (past/future). Now, let me brag: Can you get something more flexible then Proteus? 😊

#### Trial key

This is a special certificate that allows the program to work for a limited period, then switch to Demo mode. You can set this period to *any* value between 1 minute and 100 years.

You can program your app in such a way that:

* In Trial mode, all or only some of the program’s modules are available.
* In Demo mode, the program's functionality can be even more limited. For example, the 'Save' button does not work anymore so the user cannot save its work until it purchased a license.

If the user is in the trial period and he downloads a new trial version and this new trial version has a new key, you can decide if Proteus will use the new Trail certificate (get a new full trial period) or stick to wherever time was left onto the current certificate. For this just set the UpgradeToNewTrialKey field to True. Again: maximum flexibility for the programmer.

#### Default key

Any program must be delivered with a default certificate.

Usually, this will be a Trial certificate.

#### Modules

You can compartmentalize your program into "modules" - blocks of code that perform certain functions. For example, if your program converts BMP files to JPG or PNG, one module could be the BMP to JPG converter, and the other module could be the BMP to PNG converter.

The user can choose to purchase only one module or both.

The key we will send to that customer will unlock only the modules purchased.

Proteus allows up to 16 modules. See RCertificate.ModulesMask .

## Advanced topics

## Under the hood

At startup, Proteus checks if any certificate is already installed. There could be multiple certificates installed, but only one is the active/current certificate.

If a certificate is found, it is decoded and loaded into the 'CurCertif' field. Then Proteus sets the LicState field accordingly (demo, trial, registered) to the current certificate loaded.

If no certificate is found, Proteus uses the default certificate embedded within the program. Usually, this is a “Trial” certificate.

At startup, the program also checks if the system date has been tampered with. TProteus.LastSeen field is used for this. Proteus can detect if the user tries to reset the Trial period by setting back computer’s clock. An even is triggered by Proteus in this case. You can write code to decide what you do in this case (lock down the program, etc).

## Setting up your app

Step by step guide

#### Build a Key Generator

1. Start a new Delphi VCL app.

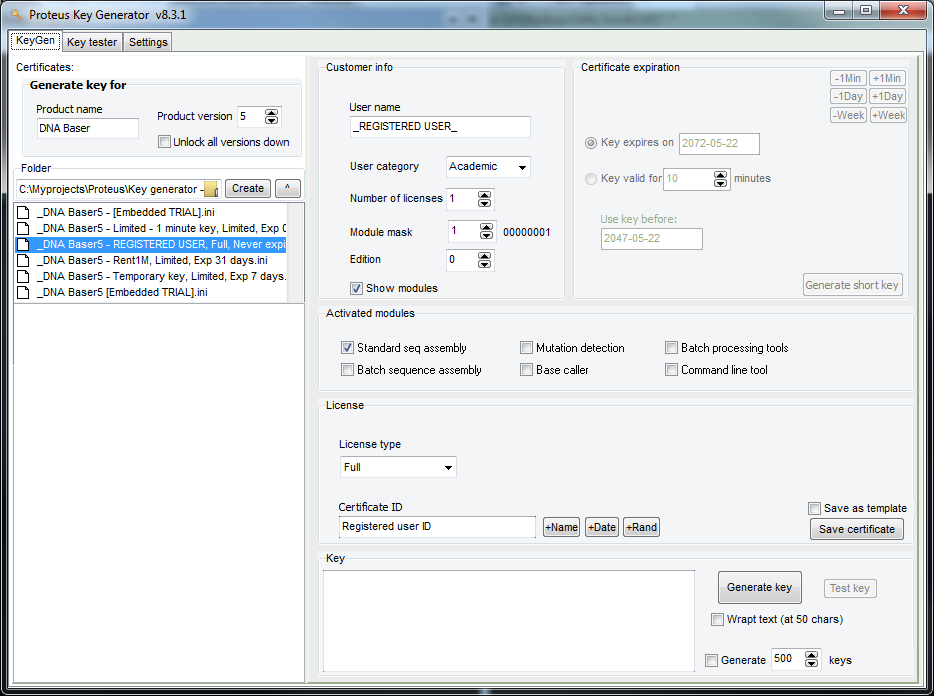
2. Drop the TProteus control on your form.

3. Click the Proteus and prepare a Trial certificate:

* Set the 'ProductName' field to something like 'My product' (or whatever).
* Set the 'Proteus.Certificate.CertificateID' field to anything we want. For example: set it to 'Trial'. If we release a new version and we want to offer a new trial period so the customers can 'taste' the product one more time, we can change the ID to something like 'Trial 2'.

Optionally Generate also a “Permanent license” certificate:

* For a Permanent certificate to be delivered to your customers (as a text key): set the ID field to something like 'Definitive key' or 'Rent' plus customer/company name (or whatever relevant info).



Example of Key Generator app built for my needs.   
Yours does not necessarily have to be this complex.

2. Your commercial app

Once your application is ready to be delivered to the customers,

* Drop a TProteus control on your main form.
* Set the 'ProductName' to the same value as in your KeyGenerator app (above).
* Set the 'AutoStoreCertificate' to True to automatically save the certificate to the storage area.
* Set the 'ObfuscateReg' to True for higher protection or to False for easier debugging.
* Enter the certificate generated with the Key Generator app above in the 'TProteus.DefaultKey' field.
* Add code in the 'OnSwitchToDemo' event handler.
* Finally, call “Proteus.Initialize” at your app startup.
* If we offer a time-based trial:
  + Call 'CheckComputerClock' from time to time to make sure the user didn't set the system clock back.
  + Use the ShowRemainingTime function to display (could be in the form's Caption) what's left from the trial period.

## Stolen keys

StolenKeys is a resource file that is compiled into the program. If one of the license keys that we sent to your customers was leaked on the Internet (on a website that delivers serials/cracks) we can put it in this list, recompile the program and release the update.

Proteus checks the current certificate against the 'StolenKeys' list at each startup.

If the current certificate is found in the Stolen Keys list, Proteus will permanently switch to Demo mode. The user is not allowed to enter any more keys (even if he has a valid key).

**How to add a 'Stolen keys' list to your project?**

This step is optional but we can do it for fun, to see how it works.

Generate one extra key. We would consider that this key was “leaked”.

1. Put the leaked key in 'CheiFurate.txt', one per line.
2. Create a new file called 'ResurseIncluse and write this inside:

Blacklist RCDATA CheiFurate.txt

1. Bind the RC into your app by adding the following line, right on top of your DPR file:

{$R 'ResurseIncluse.res' 'ResurseIncluse.rc'}

When we add a leaked key to the 'CheiFurate.txt' list, use the following format:

CertificateID=UserName

Be careful not to use spaces around the '=' sign.

Now if your try to enter that key into the program, Proteus will recognize it as a “leaked key” and refuse it.

#### Amnesty key

This allows a program that was forcefully put in Demo mode (because of a stolen key), to enter a valid key.

To do so, the user will have to enter a “secret” keyword as a key. You will find the keyword in the source code. See the "ShowEnterKeyBox" function.

After this, the user can enter the valid license key to unlock the program.

## Certificate ID

Some certificates need to be unique. For example, if we send a key to a customer that purchased a license, we might want to put a unique signature in that key so we will know from whom the key is coming. This is useful if the key was leaked over the Internet (or shared with friends/coworkers).

Another usage for a unique ID, is when we generate temporary licenses for a customer. Each key must have a unique ID so the program will know which one was used up (expired) and which one is new.

However, in other cases, we want the ID not to be unique. This is the case with Trial certificates. The Trial certificate is the same for all users.

To implement this, Proteus offers the “CertificateID” field.

**Entering the same key twice**

Proteus has is an internal check that prevents the user from entering the same key twice. This prevents a user from reusing a time-limited key over and over again.

When the user enters a key, the program checks if a certificate with the same ID already exists in the system.

If it does, the new entered key is refused.

Because of this:

* When a new minor **Trial** version (ex 1.12 to v1.13) is released, the ID of the certificate delivered with this version should remain the same to prevent the user from re-downloading the trial application and using it over and over.
* When a new major **Trial** version (ex, v1.13 to v2.0) is released, the ID of the certificate delivered with this version could be changed to allow the user to trial this brand new version.
* If we want to change the time interval on the Trial key, we might want to change the certificate ID or keep it the same, depending on if we want the user to start a new trial period or not.
* When a second (or third, etc) **Rent** key is generated and sent to the customer, the ID field should be different for each new key otherwise Proteus will refuse the key (it will think that the second key is the same as the old key). We recommend we appending the current date+time to the ID field.

## Active certificate

If multiple certificates are found in the storage area, ONLY one certificate can be active.

The last certificate entered becomes active and inactivates all other active certificates.

When the active certificate expires it switches to DEMO mode but it still remains the Active certificate. It will remain like this until a new certificate is entered.

### Initialization

Proteus should be initialized at program startup be calling TProteus.initialize. If during startup Proteus shows the "Trial expired" message box, the application will halt until the user presses ok.

If we want to allow the program to continue loading even if the "Trial expired” message box is shown, we could use a non-blocking message box (see FromAsyncMessage in Delphi LightSaber library).

### Demo

### A Proteus demo program is available for download. Extra documentation can be found also in the source code (all functions are fully documented).

### Costs

The Proteus library can be purchased for a reasonable price (source code compatible with Delphi 11/Alexandria).