Elite SL

Developer Guide

v1.0

SENSELOCK
Reliable Professional Partne

Copyrights and Trademarks

The Elite SL® with its technical documentation is copyrighted (C) 2010 to present by Senselock Software Technology Co., Ltd (Senselock). All rights reserved.

All products referenced throughout this document are trademarks of their respective owners.

All attempts have been made to make the information in this document complete and accurate. Senselock is not responsible for any direct or indirect damages or loss of business resulting from inaccuracies or omissions. The specifications contained in this document are subject to change without notice.

Contact

Senselock Software Technology Co., Ltd
Rm.1201, Blk. B, ZhuCheng Building,
ZhongGuanCun South Street, HaiDian District,
Beijing
P.R.China

Tel. +86-10-51581366
Fax +86-10-51581365

E-mail: **sense@sense.com.cn**
Website: **www.sense.com.cn**

Developer’s License Agreement

PLEASE READ THIS AGREEMENT CAREFULLY BEFORE USING THE CONTENTS THEREOF AND/OR BEFORE DOWNLOADING OR INSTALLING THE SOFTWARE PROGRAM. ALL ORDERS FOR AND USE OF THE Elite AND/OR Elite SL FAMILY PRODUCTS (including but not limited to the Kit, libraries, utilities, diskettes, disc, Senselock® and/or Senselock® Elite keys, the software component of Senselock Elite and/or Elite SL and the Elite SL License Guide) (hereinafter “Product”) SUPPLIED BY Senselock Software Technology Co., Ltd (hereinafter "Senselock") ARE AND SHALL BE, SUBJECT TO THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT.

This document is a legally binding agreement between you (either an individual or an entity) and Senselock®. If you are not willing to be bound by the terms of this agreement, you should promptly (and at least within 3 days from the date you received this package) return the unused developer's kit and the programmer's guide to Senselock. Use of the software indicates your acceptance of these terms.

GRANT OF LICENSE

The software of the Product is being licensed to you, which means you have the right to use the software only in accordance with this License Agreement. You may (a) copy the software for internal use, (b) modify the software for the purpose of integrating with your application and (c) merge the software with other programs.

NON-PERMITTED USES

Except explicitly permitted in this License Agreement, you may not (a) copy, modify, reverse engineering, decompose, assemble the Product in whole or in part, or (b) sell, lease, license, transfer, distribute all or part of the Product or rights granted in this License Agreement.

LIMITED WARRANTY

After the date of purchase, Senselock provides 12-month warranty that the Senselock Elite SL key has no material and manufacturing defects substantially. All the responsibilities of Senselock Software Technology Co., Ltd and all the compensation you can get under warranty are: you can require replace/repair the Product or accept other remedial measures.

LIMITATION OF LIABILITY

Under any circumstances, Senselock will NOT be liable for any damages arising out of usage or inability of the Product, including but not limited to: loss of data, loss of profits, and other special, incidental, joint, secondary or indirect loss.

Except for the limited warranty offered to the original buyer, Senselock is not responsible for providing any insurance to anyone on the product, performance and service including merchantability and fitness for a particular purpose.

The entire product, including Senselock Elite SL, the software, the document, other material shipped as accessories, and backups made by you are copyrighted by Senselock Software Technology Co., Ltd.

TERMINATION

Your failure to comply with the terms of this License Agreement shall terminate your license and this License Agreement.

Contents

Chapter 1. Overview 1
What is Elite SL? 1
What is Elite SL Developer Package? 1
How to Get the Most from This Guide 1
Conventions Used in This Guide 1
Document Improvement 2
Chapter 2. Features 3
Quick Binding Implementation 3
Flexible License Management 3
Secure and Convenient Remote Update 3
Sophisticated Hardware 4
Chapter 3. Definitions 5
Operation Mode 5
Administration Password 5
Developer ID 6
Product ID 6
Serial Number 6
License Module 6
Chapter 4. Installation 8
Supported Platforms 8
Supported Language 8
Toolkit Software 9
Installing the Software on Windows 9
Installed Directories and Files 10
Chapter 5. Start to Use 11
Deployment Procedure 11
Demonstration 12
Chapter 6. Using Elite SL 17
Accessing the Dongle 17
Use of AES encryption/decryption 17
Remote Update 18
Important Notice 19
Chapter 7. Elite SL Keys 20
Elite SL Master Key 20
Elite SL End-User Key 20
Chapter 8. Elite SL Toolkit – Editor 22
Initializing Elite SL 23
Initiating Protection Scheme 24
Setting License Modules 24
Setting Memory Blocks 26
Viewing device information 27
Producing Dongles 28
Producing Code Table 29

Specifying a Current Device 30
Use of License Generator tool 30
Chapter 9. Elite SL Toolkit – License Generator 31
Generating Activation Code 32
Generating Update Package 34
Generating an Synchronization Code 36
Chapter 10. Elite SL Toolkit – User Kit 37
Updating an End-User Key 38
Collecting the Information of Key 38
Chapter 11. API References - Functions 39
SLEnum 39
SLOpen 40
SLClose 41
SLGetDeviceInfo 42
SLReset 43
SLLedCtrl 44
SLVerifyPwd 45
SLChangePwd 46
SLWriteLicense 47
SLQueryLicense 48
SLGetLicense 49
SLFreeLicense 50
SLWriteMemory 51
SLReadMemory 52
SLSetProductId 53
SLEncrypt 54
SLDecrypt 55
SLUpdate 56
SLGenActivationCode 57
SLGenTimeSyncCode 59
SLGenMemoryUptPkg 60
Chapter 12. API Reference – Error Code 61
Chapter 13. API Reference – Data Structure 63
SLDevice 63
SL_LICENSE 64
SL_DATE 65
SLDeviceInfo 66
SLMemAttr 68

Overview

What is Elite SL?

Elite SL is a developer kit for software protection. APIs, tools, strategy and sample programs are provided to help you integrate your software with Elite SL dongles. By means of smart card technology and license management mechanism, Elite SL provides an easy way to protect your interests.

What is Elite SL Developer Package?

- Key (Master Key, End-User Key)
- Developer Toolkit (CD)

How to Get the Most from This Guide

The Elite SL Developer Guide is designed to assist software publishers in protecting and licensing their software by using Elite SL solution. The essential background and information on how the Elite SL make the best of its performance is provided as below.

This guide is divided into X chapters.

- Overview
- Features
- Definitions
- Installation
- Start to Use
- Using Elite SL
- Elite SL Keys
- Elite SL – Editor
- Elite SL – Generator
- Elite SL – User Kit
- API References – Functions
- API References – Error Code
- API References – Data Structure

Conventions Used in This Guide

The following conventions are used throughout this document:

Convention	Meaning
Italic	Words in italic represent file names and directory names.
Bold	Words in boldface represent keystrokes, menu items, and window names and fields.
	The caution icon flags some content you have to be careful of.
	The tip icon denotes the content should be remembered.

Document Improvement

Document Writing Team dedicates to insure the accuracy and completeness of context. Your feedback will assist them to make continuous improvement on Elite SL document. Please do not hesitate to email us, **sense@sense.com.cn**

2 Features

The following topics are covered in this chapter.

- Quick Binding Implementation
- Flexible License Management
- Secure and Convenient Remote Update
- Sophisticated Hardware

Quick Binding Implementation

Elite SL binds with the software rapidly through licensing control.

- **Verification for license**
  API functions allow you to add a license obtaining procedure before the software's vital code stored in Elite SL. If failed, the software's execution will stop immediately.
- **Verification for Developer ID and Product ID**
  Software developer is able to check out the Developer ID and Product ID stored in Elite SL that effectively forbids the illegal use, for any Developer ID or Product ID is traceable.

Flexible License Management

By means of the flexible license management, Elite SL enables software developers to make their sales schemes more varied than before.

- **Support up to 256 license modules**
  255 of license modules can be customized to manage the use of features in the software.
- **License modules can be customized**
  Licenses can be specified to control the software: of the trial days, of expiration date or of execution of features, etc.
- **Support the security algorithm**
  Each license module provides an individual key for the international standard 168-bit AES algorithm, which can be used to encrypt or decrypt data. These keys will be changed when the Product ID or Developer ID is changed.
- **Support remote update**
  License modules can be updated remotely.

Secure and Convenient Remote Update

The remote update of Elite is an off-site updating solution. Users can update license modules or data blocks of Elite SL End-User Key, without mailing the key back to software developers.

- **Encryption of the update data**
  The data is encrypted by TDES and RSA algorithm and is shielded from any forging

and tampering act. Furthermore, the update data has only one-off validity to a specific Elite SL end-user key.

- **Hardware-based production of update data**
  When producing the update data, an Elite SL master key is essential. Only end-user keys that have the same Developer ID and Product ID can be approved to use the data generated by the master key.

Sophisticated Hardware

Elite SL adopts 16-bit smart card technology. With the security level up to CC EAL4+, the chip is hardly to be cloned like other ordinary products; its powerful computing capabilities and the acceleration engine for international standard algorithms (RSA and AES, etc.) completely fulfill the demand of high security.

- **16KB large-capacity usage space**
  Elite SL allocates 8KB to read-only and read-writable data blocks respectively that greatly enriches the encryption schemes for software developers.
- **Password lockout**
  Each Elite SL has a 16-byte administration password, and the chip has a self-locking mechanism to resist the dictionary attacks. If the attack frequency exceeds the initialized value, the chip will be self-locked and no one can undongle it.
- **Globally unique serial number**
  Each Elite SL has a 64-bit serial number released by the Global Organization for Standardization instead of chip manufacturers, which is globally unique and unchangeable. By using it, you can conduct the management of software release, update and verification.
- **Hardware random number generator**
  The hardware random number generator in compliance with FIPS 140 standard (Federal Information Processing Standard formulated by the U.S. National Institute of Standards and Technology) can really achieve the generation of random numbers without rules to follow and be used to confuse the decryption person to the maximum extent.
- **Driverless**
  In full compliance with USB-HID device interface specifications, Elite SL dongle can be used in most Windows® systems without additional device drivers reducing the difficulty of deployment and maintenance.

3 Definitions

The following topics are covered in this chapter.

- Operation Mode
- Administration Password
- Developer ID
- Product ID
- Serial Number

Operation Mode

Elite SL has two operation modes, namely developer mode and end-user mode. In the developer mode, the crucial data stored can be modified. In the end-user mode, only the data stored in read-writable memory block can be modified. Please check out the following table for the privilege differences between both modes.

Operation	Developer mode	End-user mode
Modify the password	★	-
Set the product ID	★	-
Set the license modules	★	-
Set the read-only memory block	★	-
Set the read-writable memory block	★	★
Read the device information	★	★
Read the license information	★	★
Read the read-only memory block	★	★
Read the read-writable memory block	★	★



When accessing Elite SL in the protected software, please do not get into developer mode.

Administration Password

Each Elite SL has a 16-byte administration password which is initially set to “abcdefgh12345678” (between quotation marks). By default, Elite SL works in the end-user mode. By entering the correct password, Elite SL goes into developer mode.



- Do not use the initial password.
- Do not use any API functions that need the password in the protected software.
- Do not lose the password; otherwise, even the manufacturer is unable to regain or reset.

Developer ID

A Developer ID is applied to distinguish Elite SL from different software developers. When accessing Elite SL in your software, you need to provide the specified Developer ID.

It is an 8-digit hexadecimal number, consists of two 4-digit segments. The first segment is unique to identify distributors and is set by the Elite SL manufacturer. The second segment is uniquely set by the distributor to identify developers.

Product ID

A Product ID is assigned to distinguish Elite SL used by different software products of a software developer. When accessing Elite SL in your software, you need to provide the specified Product ID.

The Product ID is not initialized but generated from a seed data which is stored in the Elite SL. The seed data that interrelates to the Developer ID is 16~30-byte long, so that enumerating all the possible value is unlikely to achieve for any crackers.



- Do set the Product ID before using.
- Be cautious of 128 (Maximum modification times of Product ID).
- Keep the seed data secure.

Serial Number

Each Elite SL has a globally unique device serial number that is issued by the International Organization for Standardization, and is solidified in the chip at the time of being made. It is unchangeable.

In most cases, a Serial Number is represented as 2 hexadecimal numbers with a hyphen inbetween, such as 31323334-35363738. In the API, the number before hyphen (31323334) is named **SerialHi**, and the number after hyphen (35363738) is called **SerialLo**.



When operating remote update, the Elite SL user must provide Serial Number to the agent in charge of generating the remote update package.

License Module

A license module is a record that contains expiration date, trial days and execution count information. It can be used to control the usage of a function in your software. Each Elite SL End-User Key has 256 license modules. These license modules are numbered from 0 to 255. The first license module, License [0], is already in use; it is

permanently valid and cannot be modified. All of the remaining 255 license modules are editable and initially inactive.

The expiration date of a license module can be controlled from three aspects: Execution Count, Trial Days and Expiration Date. Besides, the license module can also be set as usable with no restrictions.

Here are some comments about how to use the license modules.

- The Trial Days and Expiration Date cannot be simultaneously specified.
- The Trial Days and Expiration Date are both using the UTC (Coordinated Universal Time) time. The calculation of the Trial Days starts when the SLGetLicense is called for the first time to access this license module.

- When specifying the Execution Count or Trial Days, the range of valid values is 0-65535. If 0 is specified, it means that the license module has already expired.

- If the Execution Count is used to control the validity of a license, for each time you call the SLGetLicense, this value will minus 1. After it is reduced to 0, calling the SLGetLicense at the next time will fail with an error code SL_LICENSE_EXPIRED returned.

- If the Trial Days are used to control the validity of a license, for each time you call the SLGetLicense, the remaining days will be calculated and updated. After the value was reduced to 0, calling the SLGetLicense at the next time will fail with an error code SL_LICENSE_EXPIRED returned.

- If the Expiration Date is used to control the validity of a license, for each time you call the SLGetLicense, the dongle will check whether it is expired. If the authorization has already expired, the SLGetLicense will fail and return the error code SL_LICENSE_EXPIRED.

- The dongle keeps monitoring the date and time variation on the computer. If the time on the computer is adjusted later than the internal time, such as from 11am to 10am, the license module failure will be caused. If this case occurs, unless the computer time is restored or the software developers provide a Synchronization Code to adjust the internal time, the module will be unavailable for a long time, usually, 5 times as much as the time difference.

Installation

4

The following topics are covered in this chapter.

- Supported Platforms
- Supported Language
- Toolkit Software
- Installing the Software on Windows
- Installed Directories and Files



Before you can begin protecting your applications, you must install the Elite SL Developer's toolkit.

Supported Platforms

Toolkit

The Elite SL software in kit can run on the following operating systems: Windows® 2000 and later Windows® Systems.

Toolkit software supports multi-language interfaces, but only English version is available at present. Other language packages can be supplied as required.

Dongle

The Elite SL dongle can run on the Windows®, Mac OS® and Linux in accordance with the USB specification. No driver is required.

API

Currently, Elite SL API can only run on the Windows platforms. API library includes:

File Format	What it means
DLL	for VB, PB, etc.
LIB	for MS Visual Studio 6.0, 2005, 2008
DCU	for Borland Delphi 7

Supported Language

Elite SL Editor and License Generator support multi-language interfaces, however currently only English version is released.



The default language is English. A specified language package for the developer toolkit can be supplied as required.

Toolkit Software

Elite SL Editor
Elite SL Editor is designed to help set up license modules and memory blocks.

Elite SL License Generator
Elite SL License Generator is used in producing the licensing data for update.

Elite SL User Kit
Elite SL User Kit is built to update Elite SL End-User Keys remotely.

Installing the Software on Windows



Before you begin installing Elite SL Software in Kit, be sure to save and exit all currently running applications.

1. Insert the Elite SL CD in an appropriate drive.
2. Access the CD root and double click *SenselockSL_v1.0.exe*. The welcome screen appears.
3. Click **Next** in the welcome screen. The License information screen appears.
4. After reading the license agreement, tick **I accept the agreement** and click **Next**. The **Select Destination Location** dialogbox screen appears. You either accept the default location of c:\SenseLock\Elite SL v1.0 or choose a different location.
5. To change the installation directory, click **Browse**.
   The **Choose Directory** dialogbox will appear.
   Navigate till you reach the location you want, and then click **OK**.
6. When you have chosen the directory, click **Next**.
   The **Select Start Menu Folder** screen appears. You may stay with the default folder *Elite SL v1.0* for saving start menu shortcuts, or you can use an existing folder by click **Browse** or create a new folder.
7. When you have selected a folder, click **Next**.
   The **Ready to Install** dialogbox appears.
8. Click **Install** to start the installation.
9. Once the installation process has finished, the **Completing the Setup Wizard** dialogbox appears.
10. Click **Finish**. The Elite SL Software has been installed to your computer successfully.

Installed Directories and Files

The installation directory contains the following files and subdirectories.

Folder	Contents
Api	Elite SL API libraries and header file.
Api\include	sensesl.h
Api\x86	Libraries for use on x86 platforms.
Api\x86\delphi7	sensesl.dcu, sensesl.inc
Api\x86\dll	sensesl.dll, sensesl.lib
Api\x86\vc6	Static-link libraries for use with MS Visual Studio 6.0.
Api\x86\vc6\md	sensesl.lib. Compiled with /MD switch.
Api\x86\vc6\ml	sensesl.lib. Compiled with /ML switch.
Api\x86\vc6\mt	sensesl.lib. Compiled with /MT switch.
Api\x86\vs2005	Static-link libraries for use with MS Visual Studio 2005.
Api\x86\vs2005\md	sensesl.lib. Compiled with /MD switch.
Api\x86\vs2005\mt	sensesl.lib. Compiled with /MT switch.
Api\x86\vs2008	Static-link libraries for use with MS Visual Studio 2008.
Api\x86\vs2008\md	sensesl.lib. Compiled with /MD switch.
Api\x86\vs2008\mt	sensesl.lib. Compiled with /MT switch.
Demos	Demo on how to use Elite SL APIs. A separate readme file is in its subfolder.
Demos\delphi7	Demo for Delphi programmers.
Demos\vs2005	Demo for VC programmers.
docs	Documentation.
tools	Tools for developers.

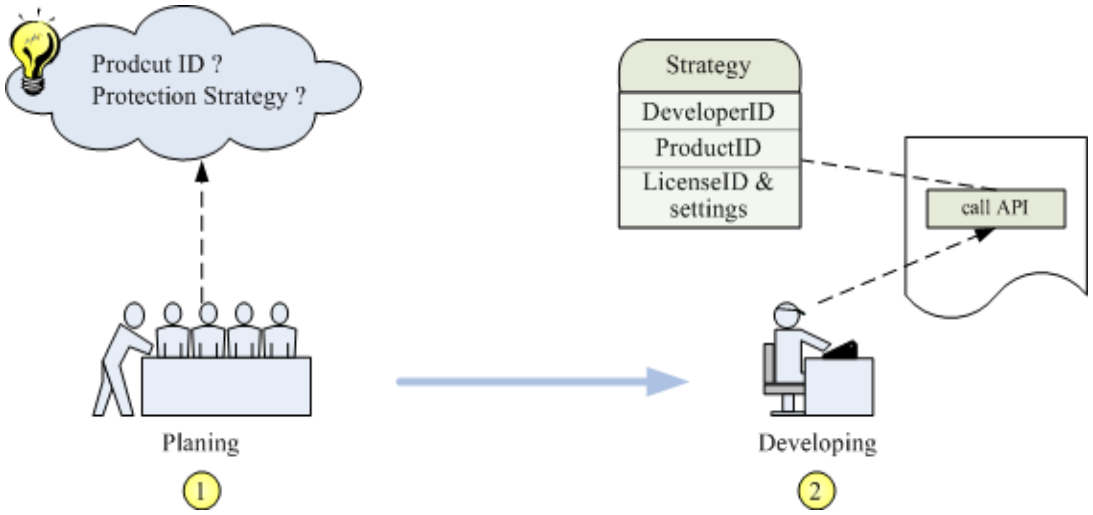
Start to Use

Deployment Procedure

When using the Elite SL to protect the software, the following several working stages are generally involved:

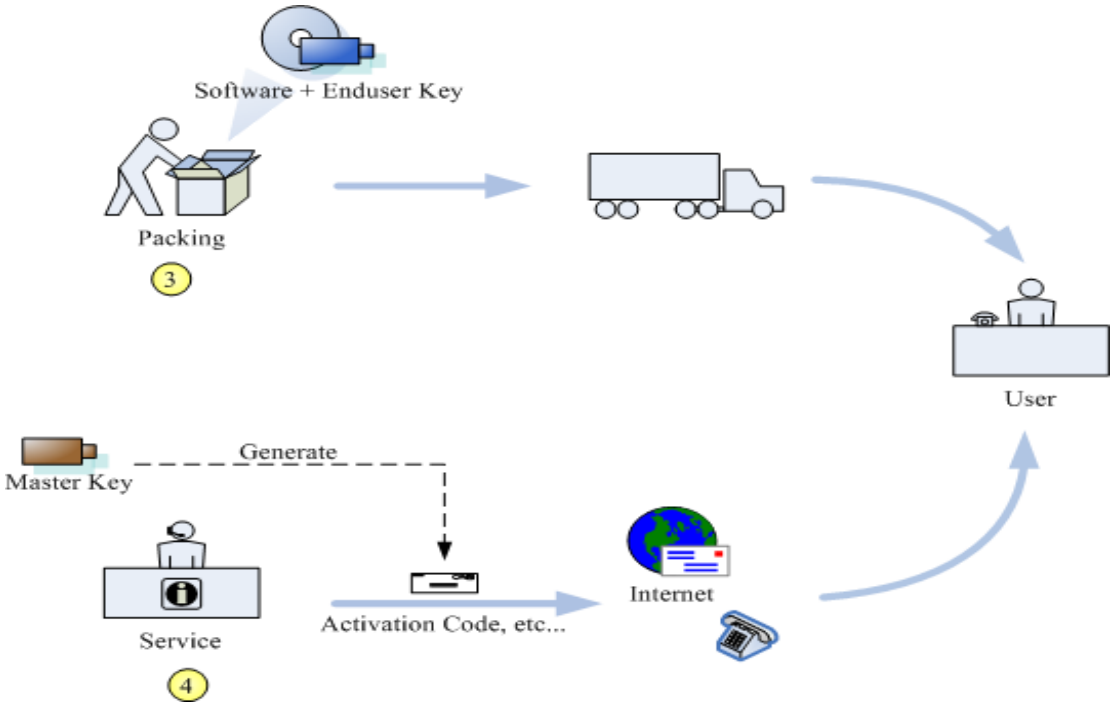
1. **Plan**
   Firstly, you should determine the keys' product ID used by the software. If more than one running application shares the use of Elite SL, you need to differentiate them via the product ID. Secondly, you need to determine the protection scheme. Specifically, it is to make a plan for license management. For example, you may use the Elite SL license modules to control the usage frequency or expiration date for specific function.

2. **Development**
   In the development stage, the developers add code to invoke Elite SL APIs to bind the Elite SL keys to the intended software.



3. **Production and Sales**
   In this phase, you must configure the dongle in accordance with the protection scheme, i.e. set the End-User Key's Product ID, Password, as well as license modules and memory block contents. Lastly, the protected software should be shipped to users with End-User Keys.

4. **Service**
   According to the protection scheme, if the remote update service is required, the service department needs to provide users with the required activation code, synchronization code or memory block update package.



Demonstration

In this section, we will introduce the Elite SL's usage procedure to you via a simple demonstration program. It is intended to help you quickly establish the overall concept and master basic usage.

- **Sample program**

  This sample program is a simple VC++ console program, its function is to display the string "Hello, world.", and then exit.

  The source code is shown below:

```
#include "stdio.h"

unsigned char szData[] = "Hello, world.";

int main(int argc, char* argv[])
{
    printf("%s\n", szData);

            return 0;
}
```

- **Designing the protection scheme**

  Here in this demo, we plan to use the Elite SL's AES encryption/decryption function to encrypt the string and store the cipher into the program. When running the program, you can call the decryption function in the End-Uer Key to decrypt the cipher, and then print out the decrypted string. In this case, we select the license module[1] for encryption/decryption.

  Initially, the dongle's license module[1] is not activated. We plan to activate it by

releasing an activation code to it and set it usable with no restrictions.

To set the Product ID, we use string “00000000000000000000000000000000” as seed.

- **Modifying the source code**
  Before modifying the original code, we need to firstly produce one End-User Key, then encrypt the string “Hello, world.” and write the cipher into the source code.

  Steps are as follows:

1. Launch the Editor tool (Tools\Editor.exe).
2. Plug the End-User Key.
3. Press the **Device Information** button to open the Device Information page.
4. After the dongle information is automatically displayed, press the Set Product ID button, then the Set Product ID dialog will pop out.
5. Enter the string “00000000000000000000000000000000” into the edit box.
6. Press the **Set** button to close this window. The Device Password dialog appears.
7. Enter the dongle password. The default password is “abcdefgh12345678” (between quotation marks).
8. Press the **OK** button. The Product ID is changed.
9. Record the value of Developer ID and Product ID.
10. In the Editor tool’s main window, press the **Project** button to open the Project page.
11. In the License Manager window, double-click the second cell in the first row. The License Editor dialogbox appears.
12. Select the **Enable this license module** checkbox.
13. Click **OK** to close the dialogbox.
14. Click **Save** to save the project file into the disk.
15. Click the **Write Key** button below the License Manager table to write the newly set license modules into the dongle. The dongle’s password is identical to the password in Step 7.
16. Double-click the second cell in the first row in the License Manager one more time, the License Editor dialogbox appears.
17. Click **Algorithm** to open the Algorithm dialog box.
18. Enter the string “Hello, world.” into the Plain Data edit box in the upper part of the dialogbox.
19. Press the **Encrypt** button in the middle of the dialogbox.
20. The decrypted cipher will be displayed in the Cipher Data edit box in the lower part of the dialogbox.
21. Click **Copy** button in the middle of the dialogbox to copy the contents of the Plain Data and Cipher Data edit box into the clipboard.
22. Paste the cipher to your source code, and rewrite as a C/C++ array, like the szCipher in Table 3-2.

Now, we have already obtained the cipher of the string “Hello, world.” and can write the Developer ID and product ID recorded in Step 9, as well as the cipher text obtained in Step 20 into the source code. Then we will add the API library and header file into the program. The modified program is as follows:

Table 3-2

```
#include "stdio.h"
#include "sensesl.h"

const unsigned long ulDeveloper ID = 0xAAAABBBB;
const unsigned long ulProduct ID   = 0x619F60DF;

unsigned char szCipher[16] = {
   0xEB, 0x33, 0x36, 0x42, 0xBF, 0x0C, 0x23, 0xC0,
   0x51, 0xC4, 0xB9, 0x0B, 0x99, 0x59, 0x1B, 0xA4
};

int main(int argc, char* argv[])
{
   ...
```



The Developer ID and Product ID should be replaced by the values you recorded.

Now, add the code to access the dongle and decrypt the cipher data. Then, print the decrypted string out. The complete code is as follows:

Table 3-3

```
#include "stdio.h"
#include "sensesl.h"

const unsigned long ulDeveloper ID = 0xAAAABBBB;
const unsigned long ulProduct ID = 0x619F60DF;

unsigned char szCipher[16] = {
   0xEB, 0x33, 0x36, 0x42, 0xBF, 0X0C, 0x23, 0xC0,
   0x51, 0xC4, 0xB9, 0x0B, 0x99, 0x59, 0x1B, 0xA4
};

int main(int argc, char* argv[])
{
   unsigned char szBuffer[16] = {0};
   unsigned long ulHandle;
   unsigned long ulRet;

   ulRet = SLOpen(&ulHandle,
            ulDeveloper ID, // specify Developer ID
            ulProduct ID,    // specify Product ID
            0,              // ignore the Serial numbers
            0,              //
            FLAG_OPEN_SHARE,
            0
            );
   if(ulRet == SL_SUCCESS)
   {
      ulRet = SLDecrypt(ulHandle,
                  1,          // specify license ID
                  szCipher,
```

```
                    szBuffer,
                    SL_AES_128
                    );
        if(ulRet == SL_SUCCESS)
        {
            printf("%s\n", szBuffer);
        }
    }
    if(ulRet != SL_SUCCESS)
    {
        printf("Error Code = %X\n", ulRet);
    }

        return 0;
}
```

Now, if the dongle we made in the previous steps is plugged into the computer, the program will print string “Hello, world.” in the console window. However, if it is replaced with a dongle that hasn’t been set for use, the program will not print out the previous string, but error code.

- **Using the Activation Code**

  When using the code in the table 3-3 with one empty dongle, the string “Hello, world.” cannot be printed out. Now, we will use the ***Activation Code*** to activate the license[1] in the empty dongle and make it work.

  Before generating and using the activation code, we firstly set the Product ID for the uninitialized End-User Key, and then prepare one Master Key. Steps are listed below.

  Steps for preparing the dongles:

1. Launch the **Editor** tool (Tools\Editor.exe).
2. Insert the uninitialized End-User Key.
3. Click the **Device Information** button to open the Device Information page.
4. After the dongle information is automatically displayed, press the **Set Product ID** button, then the Set Product ID dialogbox appears.
5. Enter the string “00000000000000000000000000000000” into the edit box.
6. Click **Set**. Then the Device Password window appears.
7. Enter the dongle password. The default password is “abcdefgh12345678”.
8. Click **OK**.
9. After the setting information is refreshed, record the value of the **Serial Number**.
10. Unplug the End-User Key and insert a Master Key.
11. Repeat step 4 to step 8 to set the Product ID of the Master Key to the same value.

  Steps for generating the Activation Code

1. Launch the **License Generator** tool (Tools\LicGenreator.exe).
2. Plug the Master Key we made in the previous steps.
3. Click the **Activation Code** button to display the page for generating activation code.
4. Set the index of **License Module** to 1.
5. Select the **Enable this license module** checkbox (selected by default).
6. Type the End-User Key’ serial number into the **Serial Number** edit box.

7. Click **Generate**. In a few seconds, the activation code will appear in the **Activation Code** edit box.
8. Keep a record of the **Activation code**.

Steps for using the Activation Code

1. Launch the **User Kit** tool (Tools\User Kit.exe).
2. Plug the End-User Key we have set already in the previous steps.
3. Select the **Update** page.
4. Write the previously recorded activation code into the **Activation Code** edit box.
5. Click **Activate** button. In a few seconds, a message box will appear committing the operation succeeded.

Now, if we plug the newly updated End-User Key into the computer, and run the routine in the table 3-3, the routine will print out “Hello world.” to inform you that activation is complete.

Using Elite SL

Accessing the Dongle

When using Elite SL dongle, the following steps are generally involved:

- Call SLOpen to open the dongle.
- Call any other API functions to access the dongle.
  Such as SLGetLicense, SLFreeLicense.
- Call SLClose to close the dongle.

When calling SLOpen, the Developer ID and Product ID should be specified, in order to prevent accessing wrong dongles. Usually, this misoperation is made by the user who is trying to access the multiple developers' software and a developer's different software with same type of dongles applied to.

In some cases, you may want to select one dongle from a group of dongles with the same Developer ID and Product ID. For example, if you would like to write your own tool program, please follow the steps below.

- Call SLEnum to enumerate all dongles.
- Call SLOpen to open the enumerated dongles one by one, and check whether they have the specified features.
- Call any other API functions to access the dongle.
- Call SLClose to close the dongle.

Use of AES encryption/decryption

AES is the latest U.S. NIST symmetric algorithm standard. It is one of the most secure symmetric key algorithms. Via the SLEncrypt and SLDecrypt function, the AES implementation of Elite SL will be triggered.

The AES in the Elite SL is bound with the license modules. When using it, you need to specify the index of a license module. When executing the encryption/decryption operation, it will obtain the key from the specified module. Each license module has an AES key. The key is related to the Developer ID and Product ID and the index of the license module. In other words, these 256 keys are unique to each developer and each product.

The AES algorithm in the Elite SL End-User Key is also related to the expiration date of the license modules. By setting the expiration date of authorization, you can control the deadline of AES algorithm.

The typical method for using AES is to encrypt the data or variables in the software. For example, you can encrypt some data in the software and save the cipher in the source code. When the software needs to use these data, use the Decrypt interface for decryption.

The **Editor** tool can be used to obtain cipher data. For example, assuming that the string "Hello, world." need to be encrypted using the key in the License[1], you can follow the steps below to do it:

1. Launch the **Editor** tool (Tools\Editor.exe).
2. Plug an End-User Key.
3. Double-click the second cell in the first row in the Project\License Manager page. The License Editor dialogbox appears.
4. Select the **Enable this license module** and click **OK**. The License Editor dialogbox is closed.
5. Click the **Write Key** button; enter the password, the license modules will be written to the dongle. If the password has never been modified, the default password is **abcdefgh12345678**.
6. Again, double-click the second cell in the first row again in the License Manager. The License Editor dialogbox appears.
7. Click the **Algorithm…** button. The Algorithm dialog appears.
8. Enter the string “Hello, world.” into the **Plain Data** edit box.
9. Click **Encrypt** button, then you can see the encrypted data in the hexadecimal data edit box in the lower part of the window.
10. Click **Copy** button to copy the original data and encrypted result to the clipboard.



Encryption and decryption require that the license module in the dongle is activated. Steps 3, 4, 5 are used to do this setting.

When using Elite SL APIs to do AES encryption/decryption, the input/output data is 16-byte long. If the data are less than 16 bytes, 0x00 is filled at the end of the data.

Remote Update

Remote update is used for modifying End-User Keys which have already been sent to software users. With the help of this technology, dongles can be securely updated without being mailed back. When executing the remote update, software developers or vendors need to use the Master Key to generate the update data.

Methods of remote update

Elite SL supports three remote update methods:

- **Activation Code**
  Activation code is a 38-byte string, which is used to change a license module in the End-User Key.
- **Synchronization Code**
  Synchronization code, a 29-byte string, is used to adjust the End-User Key’s internal timer. Usually there is no need to use the synchronization code. However, if the Trial Days or Expiration Date is restricted, for each time calling the SLGetLicense, SLEncrypt and SLDecrypt function, the dongle will check the computer time. If the time on the computer is modified later than the dongles time, such as the

adjustment from 11am to 10am, the license module will be locked. If this case occurs, unless the computer time is restored, the module will be unusable for a certain amount of time, usually, 5 times the time difference.

A Synchronization Code is used to make the dongle accept the new computer time. Releasing the synchronization code means that software development or sales staff recognizes that the user's time adjustment operation is legal, rather than for the purpose of avoiding the restriction of usage deadline. In general, the license module failure caused by the micro-adjustment (a few seconds) due to automatic time calibration can be recovered within a short time, without the need of synchronization code.

- **Memory block update package**
  Memory block update package is used to modify the read-only memory block of an End-User Key. The memory block is updated using the overall update method. Partial Updating is not supported yet.

Important Notice

When generating the Activation Code, Synchronization Code and update package, the **Serial Number** of the End-User Key must be provided and the generated data can only be recognized by this dongle.

The Master Key used to generate the update data must have the identical Developer ID and Product ID with the End-User Key. Otherwise, the generated data is not acceptable.

The activation code, synchronization code and update package can be used only once. The Synchronization Code is valid within 72 hours after its generation. UTC (Coordinated Universal Time) Time is applied to the expiration date.

The **License Generator** tool (Tools\License Generator.exe) provided by Elite SL may be used for generating the Activation Code, Synchronization Code and update package. If you need to write your own utility, please refer to:
SLGenActivationCode, **SLGenTimeSyncCode** and **SLGenMemroyUptPkg**

Elite SL Keys

The following topics are covered in this chapter.

- Elite SL Master Key
- Elite SL End-User Key

Elite SL Master Key

The Master Key is to generate update data for End-User Keys that have the same Developer ID and Product ID. It should be kept secure by software developers.

The Product ID is changeable but limited to 128 times. So we strongly recommend you to keep an individual Master key for each software product protected by Elite SL, and to set a different Product ID in it. For more details, click **here.**

Property	Description
Developer ID	A number binding to a software developer.
Product ID	A number set by software developer to specify a product.
Password	Used to secure the settings.

Properties of Master Key

Elite SL End-User Key

The End-User Key is used to store license modules and data, each copy of the software should be released with one end-user key.

The End-User Key has 256 license modules (numbered from 0 to 255) and 2 blocks of data storage. Except of License[0] which is already in use, 255 out of license modules can be customized to control the use of the software and be updated remotely like data blocks.

Each license module has an individual key for AES encryption/decryption. And this key is affected by the Developer ID and Product ID.

Property	Description
Developer ID	A number binding to a software developer.
Product ID	A number set by software developer to indicate a product.
Password	Used to secure the settings.
Serial Number	A globally unique unchangeable serial number
License Modules	Stores licenses and can be used in encryption/decryption.
Memory Blocks	Stores data.

Properties of End-User Key

Block	Address	Description
Block 1 Read-only	0x0000—0x1FFF 8192 bytes	Read only in end-user mode. Readable and writable in developer mode.
Block 2 Read-Writeable	0x2000—0x3FFF 8192 bytes	Readable and writable in end-user mode and developer mode.

Data Storage in End-User Key

Elite SL Toolkit – Editor

Elite SL Editor is designed to help set up license modules and memory blocks.

Functions

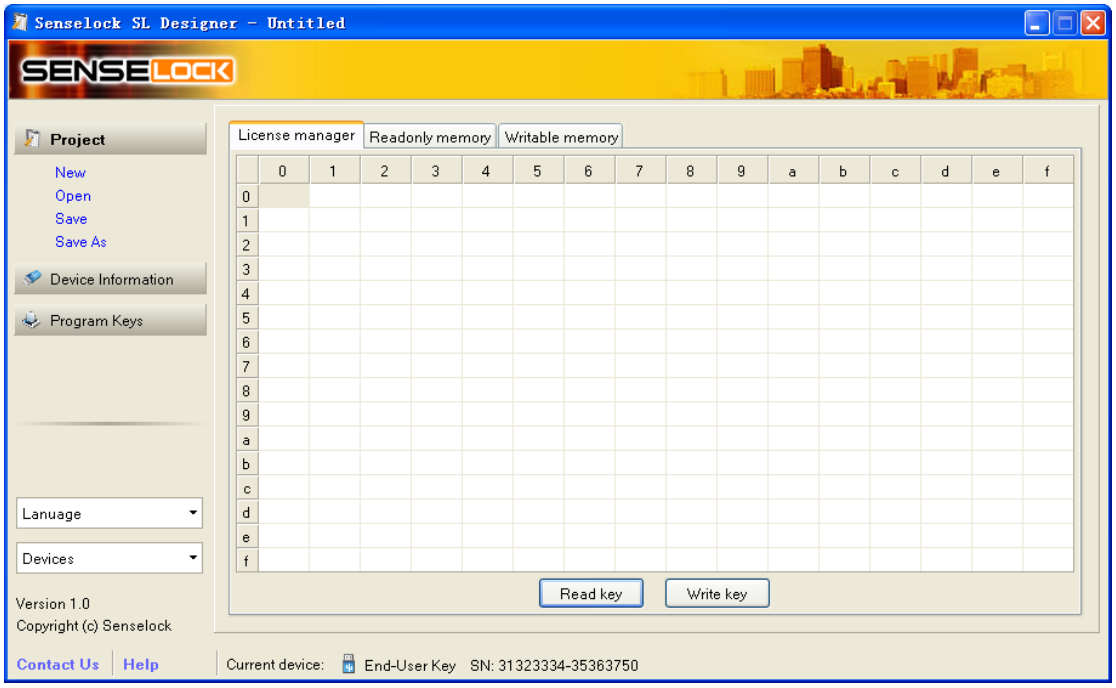
- Maintain the protection scheme which can be saved as a project file.
- View keys' hardware information.
- View all the data stored in End-User Key.
- Initialize Product ID and Administration Password.
- Download the specified protection scheme.
- Make End-User Key workable for specified protection scheme.

Opening the editor

1. From the **Start** menu, go to **Program > Elite SL v1.0**
2. Select **editor**. Its main window appears.

Interface description

When the Editor tool is launched, a window below appears.



Editor Window (with Project Stage selected)

- **Project** – Clicking this button will show the file operation buttons and a tabbed subwindow will appear in the right of the main window too. In the tabbed sub window, you can set license modules and memory blocks. Then you can save your settings into a file or just write them into a key.
- **Device Information** – Clicking this button will show a subwindow in the right of the main window, where you can see the hardware information of a selected dongle, or set the keys' Product ID and Password.
- **Program Keys** – Show a subwindow that is used to download a protection scheme

into dongles.

- **Language** – Click to change interface language.
- **Devices** – Click to select a key within a list, and set it as current device. All the subsequent read/write operations will act upon that key.
- **Contact Us** – Click to see how to contact Senselock Co., Ltd.
- **Help** – Click to view help files.

Initializing Elite SL

Elite SL Key need to be initialized before using.
Initialization procedure includes two stages: setting the Product ID and resetting the administration password.

Setting the Product ID

1. Launch the **Editor**.
2. Plug a dongle to be set.
3. In the main window, click **Device Information** button.
   The Device Information subwindow appears in the right part of the main window.
4. Click **Set Product ID** button in the device information window.
   The Set Product ID dialog box will pop up.
5. Enter the **Seed** data.
   The seed data should have 16--30 bytes. It may also be entered as hexadecimal numbers. In this case, the number of digits must be even (because each two numbers represent one byte) and 32--60 numbers requires 16--30 bytes.
6. Click **OK** button to close the dialog box.
   Then a Device Password dialogbox will pop up.
7. Enter the password. (The initial password is **abcdefgh12345678**.)
8. Click the **OK** button.
9. If the password is correct, the device information will automatically refresh within a few seconds.

The Seed used in step 5 should be kept secure and properly. You need it when setting the same Product ID into other dongles.

The Seed used in step 5 should be kept secure and properly. You need it when setting the same Product ID into other dongles.

Resetting the Administration Password

1. Launch the **Editor**.
2. Plug a dongle to be set.
3. In the main window, click **Device Information** button.
   The Device Information subwindow appears in the right part of the main window.
4. Click **Change Password** button in the device information window.
   The Change Device Password dialogbox will pop out.

5. Enter the old password and new password.
   (The initial password is **abcdefgh12345678**.)
   The password can be up to 16 bytes. It may also be entered as hexadecimal numbers. In this case, the number of digits must be even (because each two numbers represent one byte.).
6. Click **OK** button to close the dialogbox.
7. If the password is correct, the device information will automatically refresh within a few seconds.

Initiating Protection Scheme

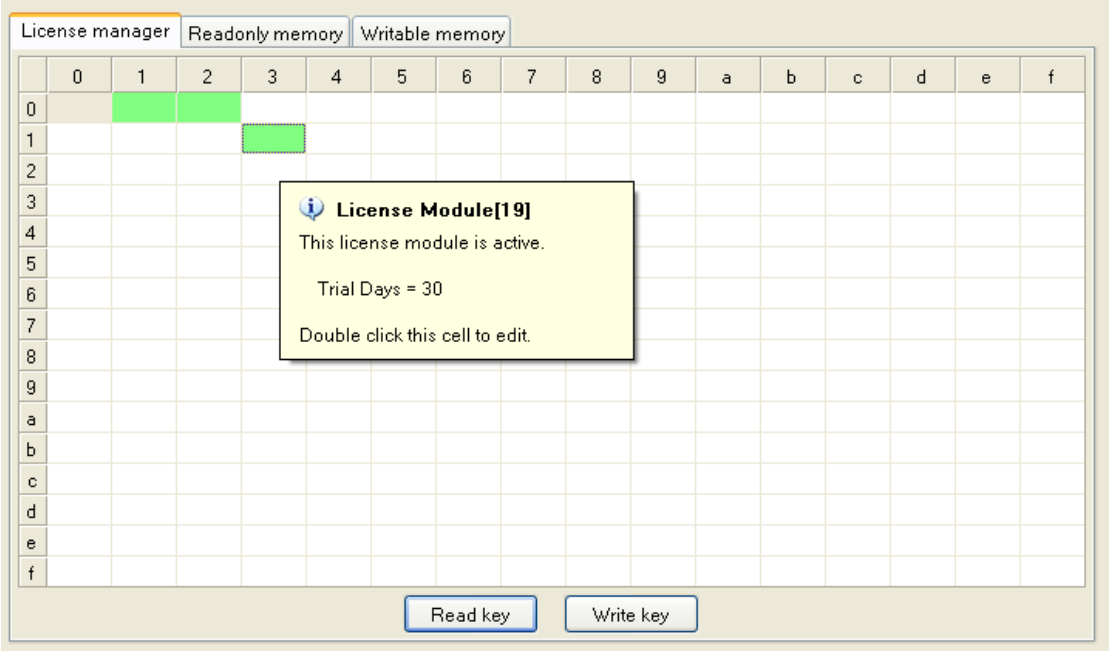
The protection scheme includes setting license modules and memory blocks. The **Project** subwindow of **Editor** is designed to help set the scheme. Furthermore, the protection scheme can be saved as a so called *project file* and loaded when you want to modify.

- **Create a new project file**
  Each time **Editor** is opened, a blank project file is automatically created for you. You can also create a new file by clicking the **New** button from the left panel of the main window.
- **Save the project file**
  You can save a project file by clicking the **Save** or **SaveAs** button from the left panel of the main window.

  When saving a file, there is a **Save with password** check box at the bottom of the **Save File** dialogbox. If it is selected, a Set File Password dialogbox will pop up. After setting the password, the file will be saved in cipher text. Person without the password cannot see the contents of the file nor download it into a dongle.
- **Open the existing file**
  You can load a project file by clicking the **OPEN** button in the left panel of the main window.
  If the file is password-protected, a File Password dialogbox will be displayed for you to input your password.

Setting License Modules

Currently, 255 license modules should be available in the newly built protection scheme except for the License[0]. Each module can be used to limit the validity of a software feature by means of **Execution Count**, **Trial Days** and **Expiration Date**. These license modules are numbered from 0 to 255, and are represented as grid cells in the **Project** subwindow, as shown below.

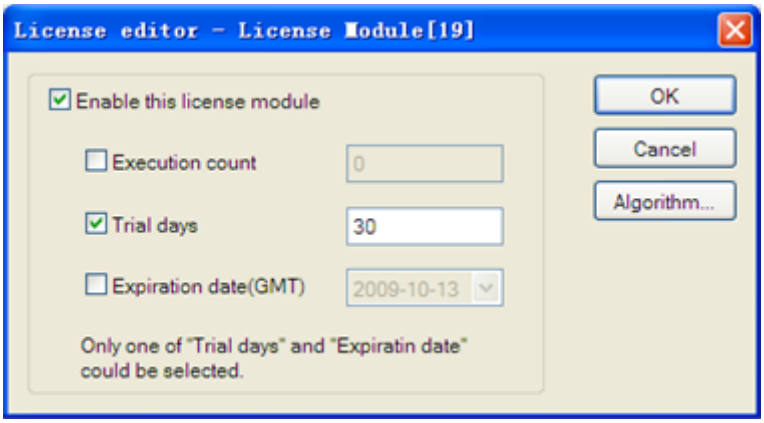


License Manager Tab

The grid style is shown above. Both the left column and top row of the table are marked with hexadecimal numbers. The combination of both numbers represents an index for a cell (decimal). For example, the serial number of the authorization module in Row 1, Column 3 is 0x13 (equivalent to 19 in Decimal).

The active modules are represented in green and inactive modules are shown in white. License[0] cannot be edited in gray.

- **View a license module**
  Move the mouse over a cell, in 0.5 seconds, the contents of the module can be displayed aside the cell.
- **Edit a license module**
  Double-click or right click in the cell, then select the License Editor from the pop-up menu. The License Editor window looks like this:



License Editor Window

- The range of Execution Count is: 0—65535.
- The range of the Trial Days is: 0—65535.
- Expiration Date uses UTC (Coordinated Universal Time). The specified date is not in the range of Expiration Date.

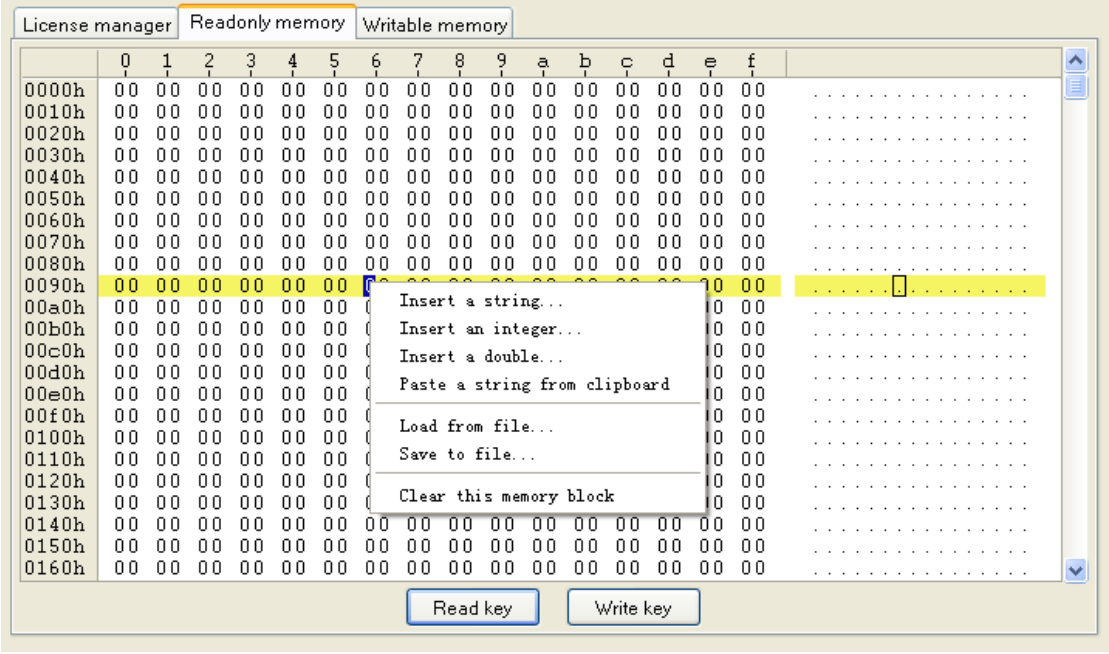
After setting up license modules, you may perform below two operations.

- **Read the license modules**
  Click **Read Key** button to read license modules from the current key.
- **Write the license modules**
  Click **Write Key** button to write license modules into the current key. This operation requires the administration password.

Setting Memory Blocks

End-User Key contains two memroy blocks which have 8192 bytes respectively. Thie first block is called Read-only Memory (0x0000--0x1FFF). The second is called Writeable Memory (0x2000--0x3FFF).

These memory blocks are represented in HEX editbox as below.



Read-only Memory Tab

- **Insert a general value**
  Type directly
- **Insert a multi-type data (string, integer, double)**
  Right click a selected location to pop out a shortcut menu.
  Click **Insert a string**, **Insert an integer** and **Insert a double**; Click Insert a string from

clipboard.

Input and output the entire block

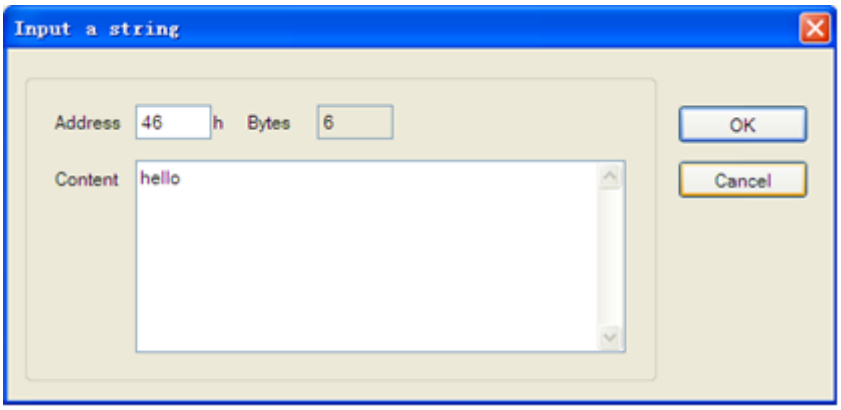
Right click a selected location to pop out a shortcut menu.

Click **Load from file**; Click **Save to file**.



When loading data from a file, the data is stored from the beginning of the block, instead of a specified location.

The string length includes its terminal NULL. As the following window shows that the string *hello* has five characters, but it occupies 6 bytes with terminal NULL added.



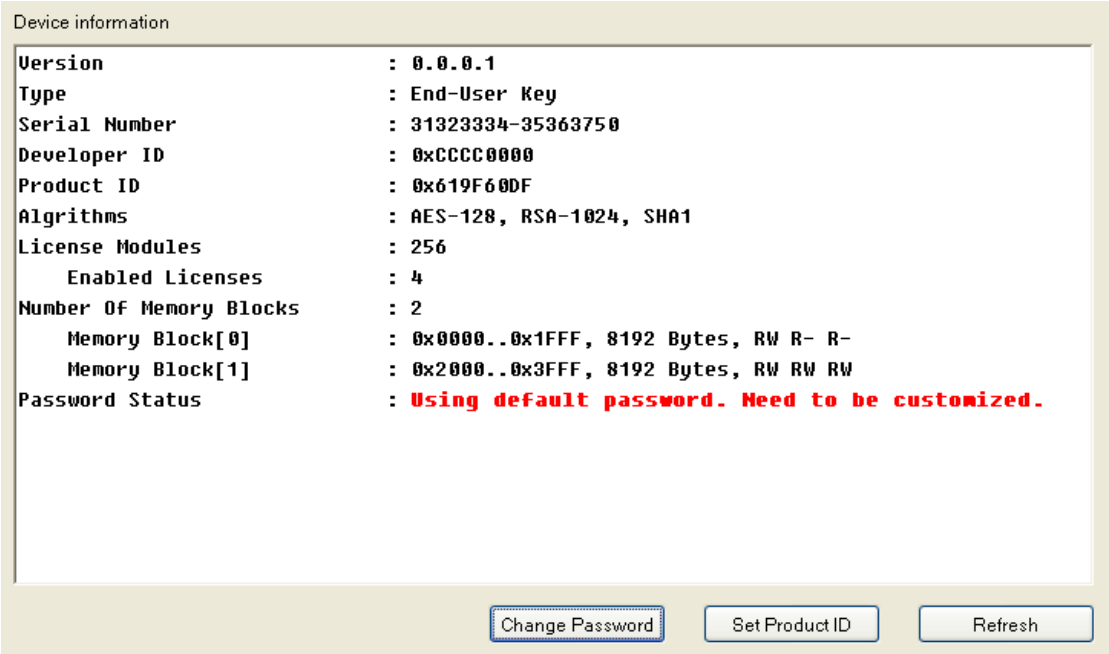
Input a String Window

When an integer is entered, the data length is fixed 4 bytes in Little-Endian format (The least significiant, or low-order byte, is stored first). The value range is -2,147,483,648 ~ 4,294,967,295.

When a floating-point number is entered, the data length is fixed 8 bytes, contains 64 bits: 1 for sign, 11 for the exponent, and 52 for the mantissa. Its range is +/–1.7E308 with at least 15-digit precision.

Viewing device information

By clicking the **Device Information** button in the left panel of **Editor**'s main window, the Device Information subwindow will appear in the right of the main window.



Press the **Refresh** button in this window, the device information of the selected dongle can be displayed. If the current dongle is a Master Key, fewer lines will be displayed.
In this window, you can also reset of the Product ID and password of dongle.

Producing Dongles

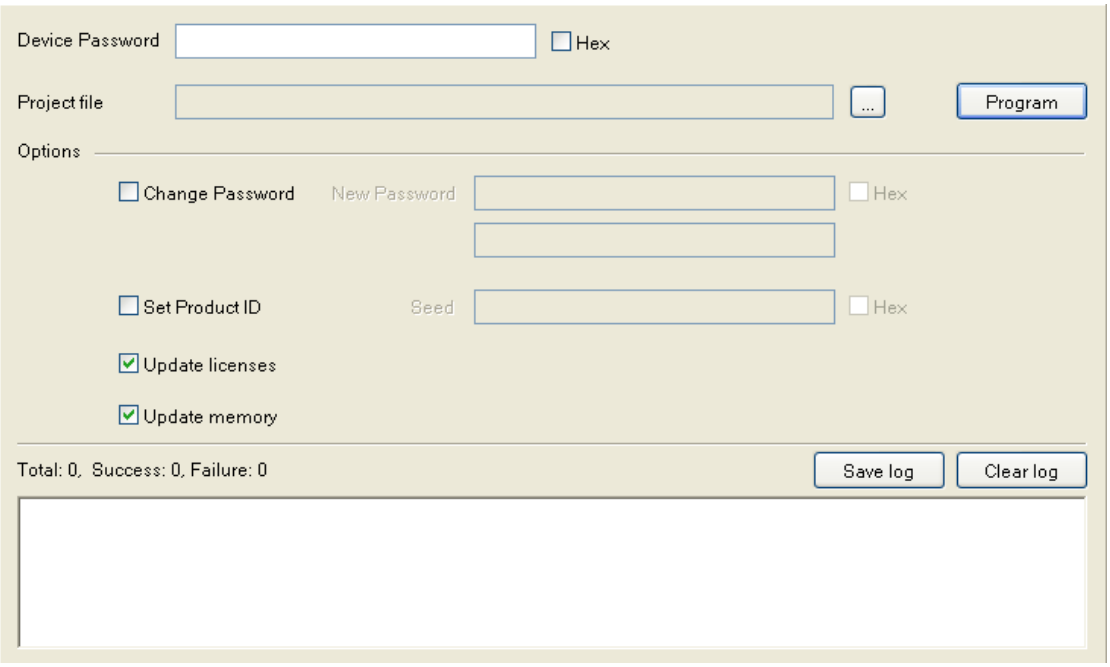
To make a dongle active for your protected software, you must download your project file into the dongle. To do this, click the **Program Keys** button in the left panel of the **Editor**'s main window. The Program Keys subwindow looks like Graph 6-7.

Program a dongle:

1. Input the password of the dongle.
2. Specify the project file which contains your protection scheme.
3. Select the **Change Password** checkbox and enter the new password.
4. Select the **Set Product ID** checkbox and enter the seed data.
5. Plug an End-User Key.
6. Click the **Program** button.
7. Repeat Step 5 and 6 to complete the production of all your dongles.
8. Press the **Save Log** button to save the production record into a disk file.



After the dongle is plugged, the system needs some time to recognize it. Therefore, if you press **Program** button and the error prompt **No Key is found** appears, the best solution is to wait for a few seconds or try it again.



Screenshot of Programming Keys

A quick way to modify the password and Product ID

The Program Keys window can also be used to modify the Password and ProdcutID of End-User Keys. The steps are listed below:

1. Enter the old password.
2. Uncheck the **Update License** checkbox.
3. Uncheck the **Update memory** checkbox.
4. Select the **Change Password** checkbox and enter the new password.
5. Select the **Set Product ID** checkbox and enter the seed data.
6. Plug your dongle. (Only End-User Key may be used.)
7. Press the **Program** button.
8. Repeat Step 5 and 6 to program other dongles.

Using this method is more applicable for continuously processing multiple End-user Keys. Compared with the method in the Device Information window, this method is faster. But it is not suitable for Master Keys.

Producing Code Table

Among the software using the Elite SL encryption dongle, in addition to the generated random numbers used in the "Challenge - response" protection mode, the seed code table method can also be adopted. The seed code table is a group of pre-prepared plain text – cipher text pairs.

A code table is a group of pre-prepared plain text – cipher text pairs. When using the code table, the protected software will call the encryption function in the dongle to encrypt a piece of plain text. If the output is identical to the expected cipher text, it means that the dongle is working properly and the copy is legal.

Currently, Elite SL End-User Key has AES encryption/decryption algorithm integrated. Moreover, the key of this algorithm is related to the Developer ID, Product ID of the dongle and the index of a license module. Therefore, the 256 license modules in the dongle provide 256 different keys. However, only the activated module can be used by the encryption/decryption function. To activate the specific license modules, you need to download your protection scheme into the dongle or activate the license module using the Activation Code.

Steps for producing the code table:

1. In the **Designer**'s main window, click the **Project** button to open the Project subwindow.
2. In License Manager, the Authorization module management page will be displayed.
3. Double-click the cell of the license module to be used.
   The License Editor window appears.
4. Select the **Enable this license module** checkbox, and then click **OK.**
5. Insert an End-User Key.
6. In the License Manager subwindow, click **Write Key** to write the license module into the dongle. This operation will ask you to input the dongle's password.
7. Double-click the cell of the license module again.
   The License Editor window appears.
8. Click **Algorithm,** the Algorithm dialogue appears.
9. Click the **Code Table.** The Code Table dialog appears.
10. In the **Count** editbox, enter the quantity of plain text – cipher text pairs to be generated.
11. Click **Generate**. The code table will be generated.

In the code table, Code_x represents the cipher text corresponding to Seed_x. Both plain text and cipher text data use two-digit hexadecimal number to represent 1-byte content. The length of Plain text and cipher text is fixed as 16 bytes. Generated code table can be saved to a file for later use.

Specifying a Current Device

Although **Editor** could make a list of all the Elite SL dongles it found, but only one of them is the current device. It means that all read/write operations are executed on this dongle. The current device is shown in the status bar at the bottom of the main window, with its Serial Number, type and icon.

To select a dongle as current device, you can click the Device button at the bottom left corner of the main window, and then select a dongle from a popup menu. In the popup, a Master Key is marked with a green icon, while an End-User Key is marked with a blue icon. When selection is finished, the status bar will be automatically refreshed to present the result.

Use of License Generator tool

Elite SL Toolkit – License Generator

License Generator is used for producing the update data.

Functions

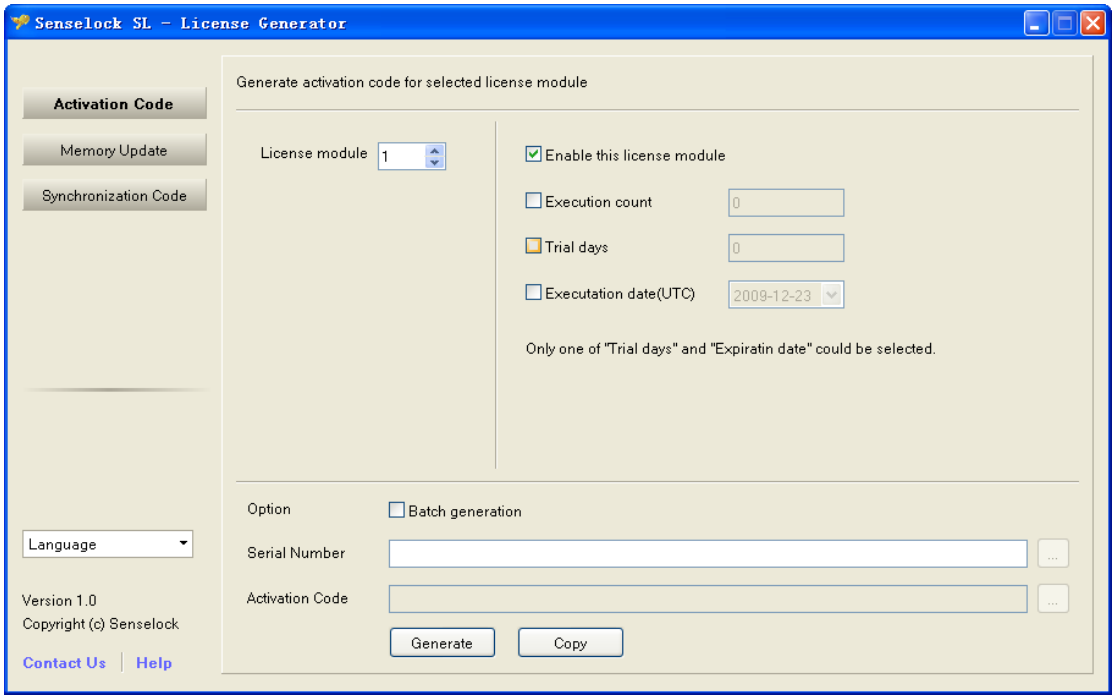
- Generate activation code
- Generate synchronization code
- Generate update package

Opening the License Generator

1. From the **Start** menu, go to **Program > Elite SL v1.0**
2. Select **License Generator**. Its main window appears.

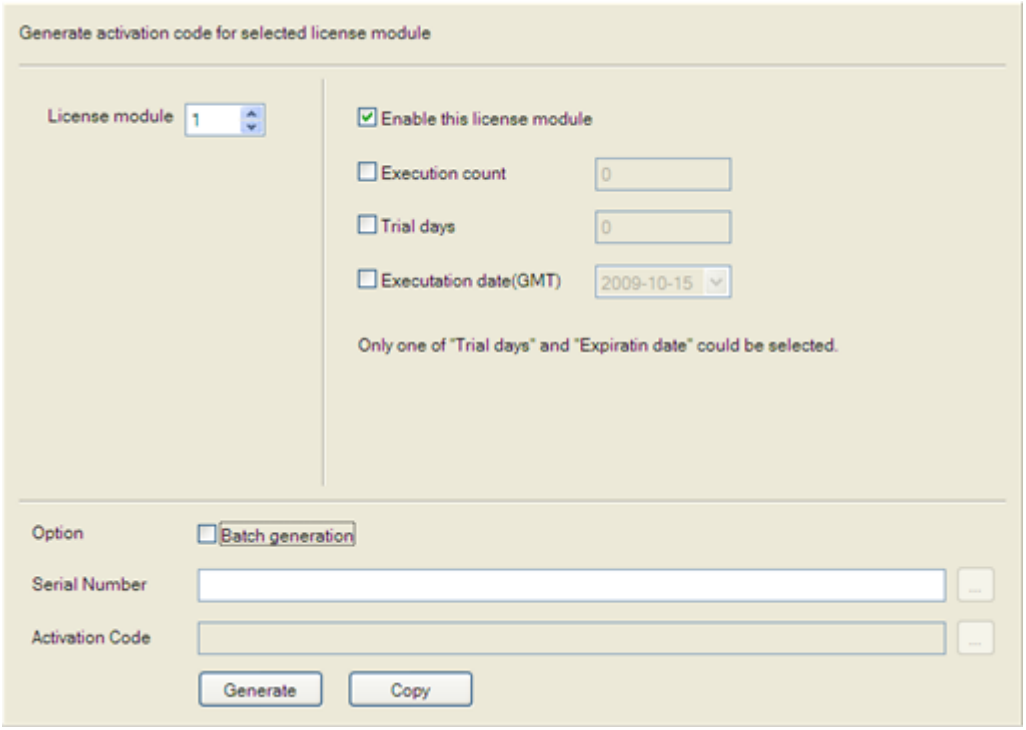
Interface description

When the License Generator tool is launched, a window like the following graph appears.



License Generator Window

- **Activation Code** – Generate the activation code for the specified authorization module in one or a group of encryption dongles.
- **Memory Update** – Generate the update package files for the read-only data blocks of one or a group of encryption dongles. Each file corresponds to one encryption dongle.
- **Synchronization Code** – Generate the synchronization code of timer in a user dongle.
- **Language** – Select the interface language.
- **Contact Us** – Click to see how to contact Senselock Co., Ltd.
- **Help** – Click to display help files.



Generating Activation Code

- **License module -** Specify a serial number of individual authorization module. Its valid range is 1-255.
- **Enable this license module -** Whether to activate the authorization module or not. If the checkbox is selected, it means that the authorization module is activated, otherwise disabled.
- **Execution count -** Specify the usage frequency of this module. The value range is 0—65535
- **Trial days -** Specify the trial days of this module. The valid range of this value is 0--65535.
- **Expiration date -** Specify the expiration date of this module. The date uses UTC time (Coordinated Universal Time). The last day is not included in the range of the expiration date.
- **Batch generation -** If the checkbox is selected, it means that the activation codes will be generated for multiple user dongles. Otherwise, the activation code will be generated for only one dongle.
- **Serial Number -** Fill in the serial number of user dongle here.
- **Activation Code -** The generated activation code will be displayed here.
- **Generate -** Press this button to generate the activation code.
- **Copy** - Used for copying the activation code to the clipboard.

- When generating the activation codes for multiple dongles, you need to specify a file using the button to the right side of the **Serial Number** editbox. This file should contain the serial numbers of dongles, and it should be a plain text file, with a line contains one Serial Number.
- When generating the activation codes for multiple user dongles, you also need to specify a file using the button to the right side of the **Activation code** editbox. This file will be used to save the generated activation codes.
- The serial number of a dongle is two 8-digit hexadecimal numbers, their input format is: hhhhhhhh-hhhhhhhh (each *h* represents one digit; the intermediate hyphen can be ignored).



- When entering the serial number, if numbers before and after the hyphen are less than 8 digits, these numbers will be considered to begin with 0. Fo example: 123-12345, it will be understood as 00000123-00012345.
- If no hyphen exists and the number has less than 8 digits, the first part of the serial number will be taken as beginning with 0, and the second part of the serial number will be taken as 00000000. For example: 1234, it will be understood as 00001234-00000000.
- If no hyphen exists and the number has over 8 digits, but less than 16, the digits after the 8th digit will be considered as the second part of the serial number and begin with 0. For example: 123456789AB00, it will be understood as 12345678-0009AB00.

Steps for generating Activation Code for one dongle

1. Uncheck the **Batch Generation** checkbox.
2. Plug your Master Key.
3. Enter the index of a **license module**.
4. According to actual needs, set the license module.
5. Enter the **Serial Number**.
6. Click the **Generate** button.
   The generated Activation Code will be displayed in the **Activation Code** editbox.

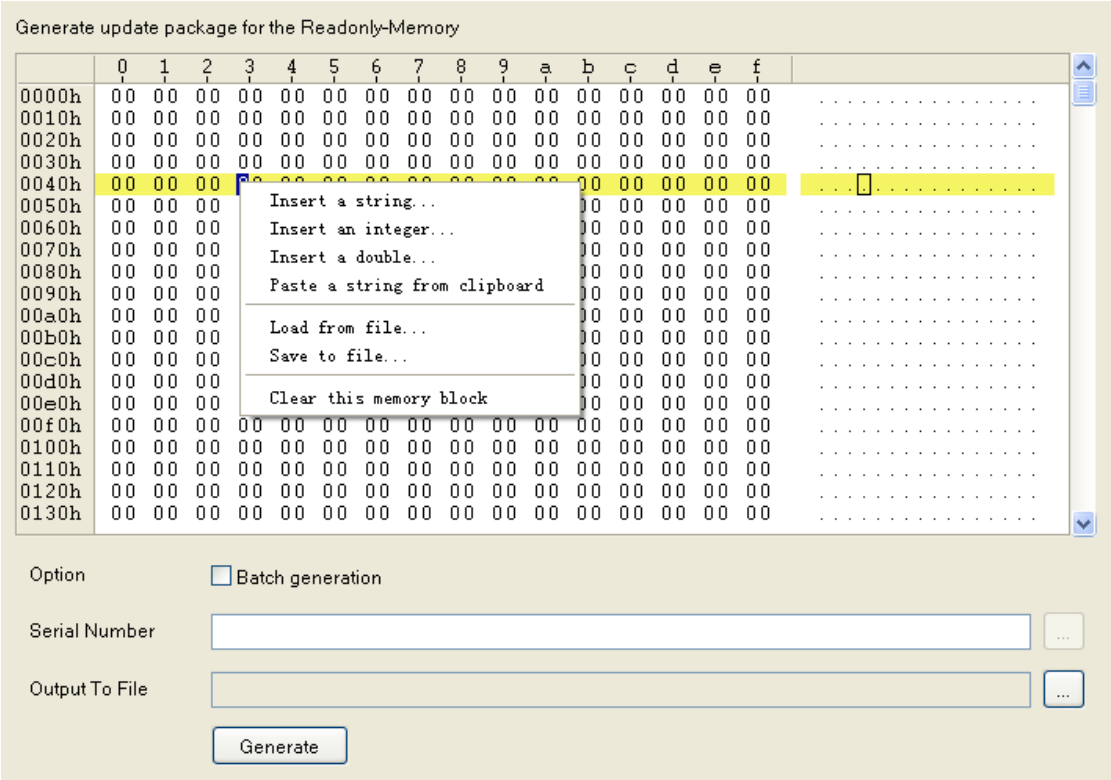
Steps for generating Activation Codes for multiple dongles

1. Check the **Batch Generation** checkbox.
   At this time, the captions of the two editboxes under the checkbox will be respectively changed to **Serial Number File**, **Output to File**. Moreover, their right-side buttons will become available.
2. Plug a Master Key.
3. Enter the index of a **license module**.
4. According to actual needs, set the license module.
5. Using the button at the right side of **Serial Number File** editbox to specify the text files containing the serial numbers of dongles.
6. Using the button at the right side of **Output to File** editbox to specify the output file.

7. Click the **Generate** button.
The generated Activation Codes will be stored in the specified output file.

Generating Update Package

The update package is used to update the Read-only Memory Block of an End-User Key.



Screenshot for generate update package

- **HEX editbox -** Used to set the content of the read-only memory block.
- **Batch generation -** If the checkbox is ticked, it means that the package will be generated for multiple user dongles. Otherwise, the package will be generated for only one dongle.
- **Serial Number -** Fill in the serial number of user dongle here.
- **Output To File –** Specify the output filename here.
- **Generate -** Press this button to generate the activation code.

- When generating packages for multiple dongles, you need to specify a file using the button to the right side of the **Serial Number** editbox. This file should contain the serial number of dongles, and it should be a plain text file, with a line containing a Serial Number.
- When generating packages for multiple dongles, you need to specify a folder by hitting the right-side button of the **Output To File** edit box. This folder will be used to store the generated package files.
- The serial number of a dongle is two 8-digit hexadecimal numbers, their input format is: hhhhhhhh-hhhhhhhh (each *h* represents one digit; the intermediate hyphen can be ignored).



- When entering the serial number, if numbers before and after the hyphen are less than 8 digits, these numbers will be considered to begin with 0. Fo example: 123-12345, it will be understood as 00000123-00012345.
- If no hyphen exists and the number has less than 8 digits, the first part of the serial number will be taken as beginning with 0, and the second part of the serial number will be taken as 00000000. For example: 1234, it will be understood as 00001234-00000000.
- If no hyphen exists and the number has over 8 digits, but less than 16, the digits after the 8th digit will be considered as the second part of the serial number and begin with 0. For example: 123456789AB00, it will be understood as 12345678-0009AB00.

Steps for generating update package for one dongle

1. Uncheck the **Batch Generation** checkbox.
2. Plug your Master Key.
3. Enter your data into HEX editbox.
4. Enter the **Serial Number**.
5. Specify a filename in the **Output to File** editbox.
6. Click the **Generate** button.
   The generated package will be saved into a specified output file.

Steps for generating update packages for multiple dongles

1. Check the **Batch Generation** checkbox. At this time, the captions of the two edit boxes under the checkbox will be respectively changed to **Serial Number File**, and **Output To Folder**.
2. Plug your Master Key.
3. Enter your data into HEX editbox.
4. Use the right-side button of **Serial Number File** edit box to specify the text file containing the serial numbers of dongles.
5. Use the right-side button of **Output to Folder** edit box to specify the folder for storing the packages.
6. Click the **Generate** button.
   Then the generated files will be saved into the specified folder. File name format:

hhhhhhhh-hhhhhhhh-yyyymmddHHMMSS.lic,
File names are made of the serial number you provided and followed by the generation time: year, month, day, hour, minute and second.

Generating an Synchronization Code

Steps are as follows:

1. Plug the Master Key.
2. Enter the **Serial Number**.
3. Click the **Generate** button.

- The serial number of a dongle is two 8-digit hexadecimal numbers, their input format is: hhhhhhhh-hhhhhhhh (each *h* represents one digit; the intermediate hyphen can be ignored).
- When entering the serial number, if numbers before and after the hyphen are less than 8 digits, these numbers will be considered to begin with 0. Fo example: 123-12345, it will be understood as 00000123-00012345.



- If no hyphen exists and the number has less than 8 digits, the first part of the serial number will be taken as beginning with 0, and the second part of the serial number will be taken as 00000000. For example: 1234, it will be understood as 00001234-00000000.
- If no hyphen exists and the number has over 8 digits, but less than 16, the digits after the 8th digit will be considered as the second part of the serial number and begin with 0. For example: 123456789AB00, it will be understood as 12345678-0009AB00.

Elite SL Toolkit – User Kit

The **User Kit** is to update for Elite SL End-User Key. It is designed to help understand the remote update mechanism, and is used to test your update scheme before you program an updating tool for your software.

Functions

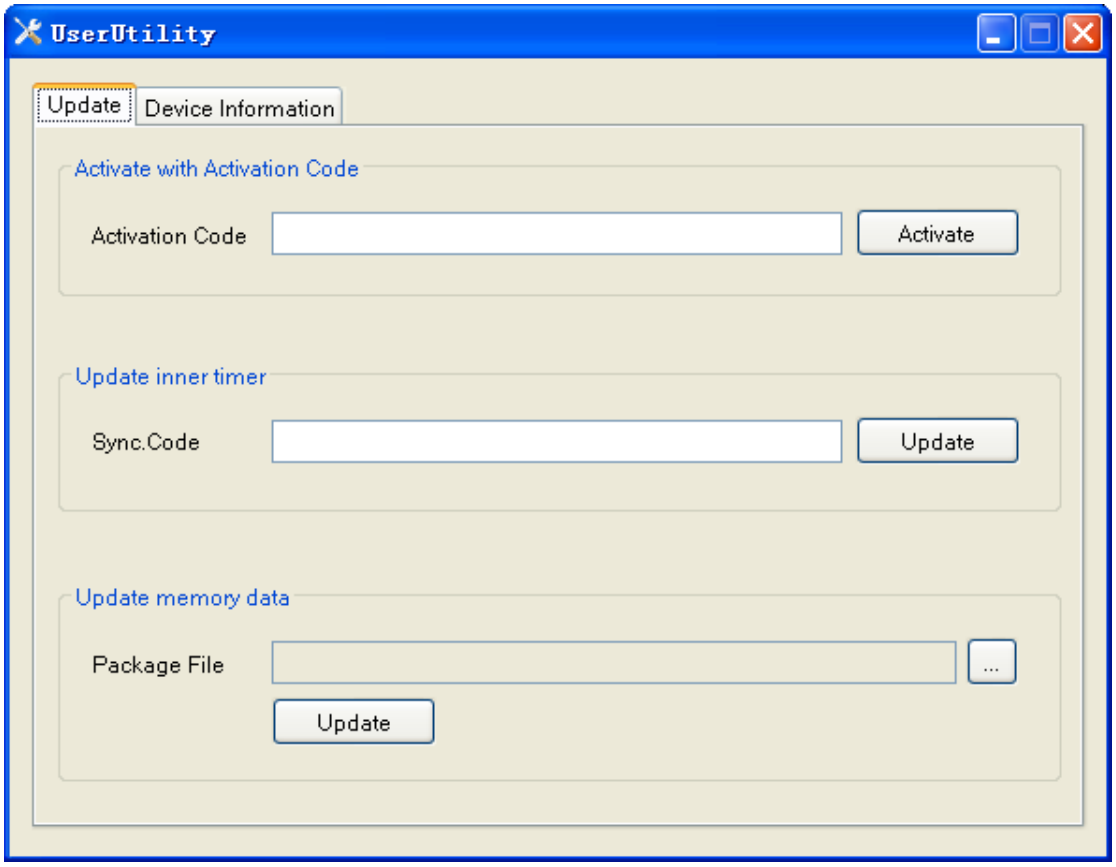
- Update End-User Key by activation code
- Update End-User Key by synchronization code
- Update End-User Key by update package
- View the device information.

Opening the User Kit

1. From the **Start** menu, go to **Program > Elite SL v1.0**.
2. Select **User Kit.** Its main window appears.

Interface description

When the User Kit is launched, a window like the following graph appears.



User Kit Window

- **Activation Code** – Enter the activation code here.
- **Activate** – Press this button to execute the update using activation code.

- **Synchronization Code** – Enter synchronization code here.
- **Update** – Press this button to execute the update using synchronization code.
- **Package File** – Browse and select an update package file here.
- **Update** –Press this button to execute using the update package file.

Updating an End-User Key

Steps for Updating from an Activation Code

1. Launch the **UserUtility** tool.
2. Plug a End-User Key to be updated.
3. Enter the obtained **Activation Code**.
4. Click the **Activate** button to complete the update operation.

Steps for Updating from a Synchronization Code

1. Launch the **UserUtility** tool.
2. Plug an End-User Key to be updated.
3. Enter the obtained **Synchronization Code**.
4. Click the **Update** button at the right of the editobx to complete the operation.

Steps for Updating from a package file

1. Launch the **UserUtility** tool.
2. Plug a End-User Key to be updated.
3. Use right-side button of **Package File** editbox to specify the name of package file.
4. Click the **Update** button to complete the operation.

Collecting the Information of Key

The Device Information page is used to collect a dongle's information. Its aim is to help developers to analyze errors and debug. The collected data can be saved in a file, which facilitates users to decide whether this information should be sent to the software developers. It can also be composed an e-mail, but the recipient needs to be filled by users.

SLEnum

Enumerate all Elite SL dongles and save the returned device information into a list,

```
SL_DWORD SLAPI SLEnum(
    SL_DWORD    ulDeveloperId,
    SL_DWORD    ulProductId,
    PSLDevice   pSLDevice,
    SL_DWORD    *pulSize,
    SL_BYTE     ucFlag
);
```

Parameters

ulDeveloperId	[IN] Developer ID.
ulProductId	[IN] Product ID.
pSLDevice	[OUT] A pointer to a buffer for storing the device information.
pulSize	[IN][OUT] A pointer a number variable. As input, it is a byte count in the buffer pointed by *pSLDevice*. As output, it is a byte count actually occupied by device information. If the buffer length is insufficient, this value is the expected buffer length in bytes.
ucFlag	[IN] Reserved; set as 0.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

If *pSLDevice* is NULL or the buffer length specified by **pulSize* is insufficient, the function will return the error code **SL_INSUFFICIENT_BUFFER**(0x80000006), and the required buffer length will be stored into a variable pointed by *pulSize*.

If the Developer ID and Product ID are set as 0, it means that all of the Elite SL dongles should be enumerated. Otherwise, only the dongles with specified Developer ID and Product ID will be enumerated.

SLOpen

Open the first matching dongle according to the specified Developer ID, Product ID and Serial Number.

```
SL_DWORD SLAPI SLOpen(
   SL_DWORD   *pulSLHandle,
   SL_DWORD   uldeveloperId,
   SL_DWORD   ulProductId,
   SL_DWORD   ulSerialHi,
   SL_DWORD   ulSerialLo,
   SL_BYTE    ucMode,
   SL_BYTE    ucFlag
);
```

Parameters

pulSLHandle	[OUT] If it is successful, the device handle will be saved into a variable pointed by it.
ulDeveloperId	[IN] Developer ID.
ulProductId	[IN] Product ID.
ulSerialHi	[IN] The first part of the Serial Number.
ulSerialLo	[IN] The second part of the Serial Number.
ulMode	[IN] Open mode.
ucFlag	[IN] Reserved, set as 0.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

If both of the *ulSerialHi* and *ulSerialLo* are 0, it accesses the first dongle that has the specified Developer ID and Product ID.

ulMode can be one of the following values:

- **FLAG_OPEN_SHARE**
  Open in the shared mode.
- **FLAG_OPEN_EXCLUSIVE**
  Open in the exclusive mode.

The above two can be combined with the following constants:

- **FLAG_OPEN_FORCE**
  Open forcibly in the specified mode with being flagged even the dongle is already occupied by other programs.

SLClose

Close the dongle.

```
SL_DWORD SLAPI SLClose(
    SL_DWORD    ulSLHandle
);
```

Parameters

ulSLHandle	[IN] The device handles of the dongle to be closed.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

If the dongle is opened in the shared mode, after executing the SLClose, this dongle's other handles will still be valid.

SLGetDeviceInfo

Obtain the dongle's device information.

```
SL_DWORD SLAPI SLGetDeviceInfo(
   SL_DWORD        ulSLHandle,
   PSLDeviceInfo   pSLDeviceInfo
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
ulDeveloper ID	[OUT] A pointer to a **SLDeviceInfo** structure.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

SLReset

Reset the dongle.

```
SL_DWORD SLAPI SLReset(
   SL_DWORD       ulSLHandle
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

No matter the dongle is opened in the shared or exclusive mode, after calling the SLReset, all of the handles will be invalid. If you want to use the dongle, you need to call the SLOpen to re-open the dongle.

SLLedCtrl

Control dongle's LED light.

```
SL_DWORD SLAPI SLLedCtrl(
   SL_DWORD   ulSLHandle,
   SL_DWORD   ulLEDStatus,
   SL_DWORD   ulVal
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
ulLEDStatus	[IN] Status of LED light
ulVal	[IN] If LED is set as flash mode, this parameter will specify the flash frequency.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The *ulLEDStatus* can take one of the following values:

- **SL_LED_ON**
  Open. The value of ulVal is ignored.
- **SL_LED_OFF**
  Close. The value of ulVal is ignored.
- **SL_LED_FLASH**
  Flash. The ulVal parameter specifies the flash frequency.

SLVerifyPwd

Verify the password. This function is supplied for the software developers to use in their own dongle setting tools.

```
SL_DWORD SLAPI SLVerifyPwd(
   SL_DWORD    ulSLHandle,
   SL_BYTE     *pucPasswd,
   SL_BYTE     ucPwdType,
   SL_DWORD    *pulRetryTimes
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
pucPasswd	[IN] Pointer to a buffer containing the passwords.
ucPwdType	[IN] Password type identifier.
pulRetryItmes	[OUT] Pointer to an unsigned long integer variable; used for receiving the remaining password retry frequencies.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The buffer pointed by *pucPasswd* should not be shorter than **SL_PWD_SIZE** bytes.

Only developers need to call this function. The password is 16-byte.long and allows retry for 16 times. If the first 8 bytes are correct, but the last 8 bytes are wrong, the frequency of retry will minus 1. If the first 8 bytes are wrong, the frequency will remain unchanged. Once the frequency reaches 0, it is unable to verify the password at all.

The *ucPwdType* can take one of the following values:

- **SL_PWD_TYPE_DEV**
  Developer's password. After verifying the developer password, you can modify the license modules and read-only memory block of the dongle.
- **SL_PWD_TYPE_USER**
  User password. Reserved. Currently, there is no need to use this identifier.



This function should not be used in the protected software.

SLChangePwd

Modify the password. This function is supplied for the software developers and used in their own dongle setting tools.

```
SL_DWORD SLAPI SLChangePwd(
   SL_DWORD    ulSLHandle,
   SL_BYTE     *pucPasswd,
   SL_BYTE     ucPwdType
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
pucPasswd	[IN] Pointer to a buffer containing the new password.
ucPwdType	[IN] Password type identifier.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The function cannot be used before initiallization of the dongle's Product ID.

Before calling the function, you need to call the SLVerifyPwd to verify the old password.

The buffer pointed by *pucPasswd* should not be shorter than **SL_PWD_SIZE** bytes.

The *ucPwdType* can take one of the following values:

- **SL_PWD_TYPE_DEV**
  Developer's password. After verifying the developer password, you can modify the license modules and read-only memory block of the dongle.
- **SL_PWD_TYPE_USER**
  User password. Reserved. Currently, there is no need to use this identifier.



This function should not be used in the protected software.

SLWriteLicense

Set the content of a license module. This function is supplied for the software developers to use in their own dongle setting tools.

```
SL_DWORD SLAPI SLWriteLicense(
   SL_DWORD     ulSLHandle,
   SL_DWORD     ulLicenseId,
   SL_LICENSE   *pLicense
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
ulLicenseId	[IN] The index of a license module. The range of values: 1--255.
pLicense	[IN] Pointer to an **SL_LICENSE** structure

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The function cannot be used before initiallization of the dongle's Product ID.

Before calling the function, you need to call the SLVerifyPwd to verify the password.

The *pLicense* should not be NULL. The structure pointed by it should have the new content for the license module.

Although the license modules are numbered form 0 to 255, however, because ulLicense cannot be modified, the valid range of indexes is 1--255.

SLQueryLicense

Obtain the content of a license module.

```
SL_DWORD SLAPI SLQueryLicense(
   SL_DWORD     ulSLHandle,
   SL_DWORD     ulLicenseId,
   SL_LICENSE   *pLicense
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
ulLicenseId	[IN] The index of a license module. The range of values: 0--255.
pLicense	[OUT] Pointer to an **SL_LICENSE** structure.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The function cannot be used before initiallization of the dongle's Product ID.

The *pLicense* should not be NULL。

SLGetLicense

Obtain the authorization.

```
SL_DWORD SLAPI SLGetLicense(
   SL_DWORD   ulSLHandle,
   SL_DWORD   ulLicenseId,
   SL_DWORD   *pulLicSesData
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
ulLicenseId	[IN] The index of a license module. The range of values: 0--255.
pulLicSesData	[OUT] Pointer to an unsigned long integer variable to receive the license ID, This identifier will be needed in releasing the license.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The function cannot be used before initiallization of the dongle's Product ID.

SLFreeLicense

Release a license module.

```
SL_DWORD SLAPI SLFreeLicense(
   SL_DWORD  ulSLHandle,
   SL_DWORD  ulLicSesData
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle
ulLicSesData	[IN] LiceseID returned by **SLGetLicense**

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The function cannot be used before initiallization of the dongle's Product ID.

SLWriteMemory

Write data into the memory block of a dongle.

```
SL_DWORD SLAPI SLWriteMemory(
   SL_DWORD    ulSLHandle,
   SL_DWORD    ulAddress,
   SL_BYTE     *pucVal,
   SL_DWORD    ulLen
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
ulAddress	[IN] Storing Address.
pucVal	[IN] Pointer to a buffer containing the data to be written.
ulLen	[IN] The length of data in bytes.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The function cannot be used before initiallization of the dongle's Product ID.

The data storage in a dongle is divided into two Memroy Blocks. Their addresses range respectively:

- **0x0000—0x1FFF, 8192 bytes, read only.**
- **0x2000—0x3FFF, 8192 bytes, read-writeable.**

When data is being written into the storage, the border cannot be crossed.

If the write operation is aimed at the read-only Memory Block, it is required that **SLVerifyPwd** should be called before this function.

If the write operation is aimed at the read-writeable Memory Block, the varification of password is not required.

SLReadMemory

Read the data from the memory block of a dongle.

```
SL_DWORD SLAPI SLReadMemory(
   SL_DWORD    ulSLHandle,
   SL_DWORD    ulAddress,
   SL_BYTE     *pucVal,
   SL_DWORD    ulLen
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
ulAddress	[IN] Data area address
pucVal	[OUT] Pointer to a buffer to receive the data.
ulLen	[IN] The length of the buffer in byte.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The function cannot be used before initiallization of the dongle's Product ID.

The data storage in a dongle is divided into two Memroy Blocks. Their addresses range respectively:

- **0x0000—0x1FFF, 8192 bytes, read only.**
- **0x2000—0x3FFF, 8192 bytes, read-writeable.**

When data is being written into the storage, the border cannot be crossed.

SLSetProductId

Set the dongle's Product ID. This function is supplied for the software developers to use in their own dongle setting tools.

```
SL_DWORD SLAPI SLSetProductId(
   SL_DWORD    ulSLHandle,
   SL_BYTE     *pucSeed,
   SL_DWORD    ulSeedLen,
   SL_DWORD    *pulPid
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
pucSeed	[IN] Pointer to a buffer containing the seed data. This seed data should have 16—30 bytes.
ulSeedLen	[IN] The length seed data in bytes.
pulPid	[OUT] Pointer to a 4 bytes buffer to receive the newly generated Product ID. Mostly, this data is treated as an integer.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

Before calling this function, you need to call the **SLVerifyPwd** to verify the password.

The dongle's Product ID is genreated from a seed data provided by a developer. The length of seed data should be 16--30 bytes. The generated Product IDs will be stored in the variable pointed by pulPid. When calling the **SLEnum** and **SLOpen** function, you can specify this Product ID to open the specific dongle.

SLEncrypt

Encrypt the data.

```
SL_DWORD SLAPI SLEncrypt(
   SL_DWORD    ulSLHandle,
   SL_DWORD    ulLicenseId,
   SL_BYTE     *pucPlainText,
   SL_BYTE     *pucCipherText,
   SL_BYTE     ucAlgoFlag
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
ulLicenseId	[IN] The index of a license module. The range of values: 0--255.
pucPlainText	[IN] Pointer to a buffer containing plain text.
pucCipherText	[OUT] Pointer to a buffer to receive the cipher.
ucAlgoFlag	[IN] Algorithm identifier. Currently, it must be set as **SL_AES_128**.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The function cannot be used before initiallization of the dongle's Product ID.

The license module specified by *ulLicenseId* must be activated.

The length of the data to be encrypted must be 16-byte. The cipher text obtained is 16-byte as well.

Currently, *ulAlgoFlag* must be set as **SL_AES_128**.

SLDecrypt

Decrypt the data.

```
SL_DWORD SLAPI SLDecrypt(
   SL_DWORD    ulSLHandle,
   SL_DWORD    ulLicenseId,
   SL_BYTE     *pucCipherText,
   SL_BYTE     *pucPlainText,
   SL_BYTE     ucAlgoFlag
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
ulLicenseId	[IN] The index of a license module. The range of values: 0--255.
pucCipherText	[IN] Pointer to a buffer containing the cipher text.
pucPlainText	[OUT] Pointer to a buffer to receive plain text.
ucAlgoFlag	[IN] Algorithm identifier. Currently, it must be set as **SL_AES_128**.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The function cannot be used before initiallization of the dongle's Product ID.

The license module specified by *ulLicenseId* must be activated.

The length of the data to be decrypted must be 16-byte. The plain text obtained is 16-byte as well.

Currently, *ulAlgoFlag* must be set as **SL_AES_128**.

SLUpdate

Use the update package or update code to update an End-User Key (dongle).

```
SL_DWORD SLAPI SLUpdate(
   SL_DWORD    ulSLHandle,
   SL_BYTE     *pucUpdateBuf,
   SL_BYTE     ucFlag,
   SL_DWORD    ulLen
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
pucUpdateBuf	[IN] Pointer to a buffer containing the update package or update code.
ucFlag	[IN] Type identifier of update data.
ulLen	[IN] Length in bytes of the update package or update code.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

The function cannot be used before initiallization of the dongle's Product ID.

The *ucFlag* can take one of the following values:

- **SL_UPT_ACTIVE_CODE**
  Use an Activation Code to execute the update for a license module.
- **SL_UPT_TIME**
  Use a Synchronization Code to adjust the dongle's internal time, so that make the dongle synchronized with the computer time.
- **SL_UPT_MEMORY**
  Use the update package to update the read-only Memory Block in the dongle.

Currently, the Activation Code is a 38-byte ANSI string (excluding the terminal **NULL**), the Synchronization Code is a 29-byte ANSI string (excluding the terminal **NULL**). The update package for read-only Memory Block is 9728-byte binary data.

SLGenActivationCode

Generate the Activation Code. This function is supplied for the software developers to write their own authorization tools.

```
SL_DWORD SLAPI SLGenActivationCode(
   SL_DWORD     ulSLHandle,
   SL_DWORD     ulSerialHi,
   SL_DWORD     ulSerialLo,
   SL_DWORD     ulLicenseId,
   SL_LICENSE   *pLicense,
   SL_BYTE      *pucActivationCode,
   SL_DWORD     *pulSize
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle
ulSerialHi	[IN] The first part of the Serial Number.
ulSerialLo	[IN] The second part of the Serial Number.
ulLicenseId	[IN] The index of a license module. The range of values: 1-255.
pLicense	[IN] Pointer to an **SL_LICENSE** structure.
pucActivationCode	[OUT] Pointer to a buffer to receive the activation code.
pulSize	[IN] [OUT] Pointer to an integer variable. As input, provides the length in byte of the buffer pointed by *pucActivationCode*. As output, stores the length of the Activation Code.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

This function is only applicable to the Master Key.

Although the license modules are numbered from 0 to 255, however, because license[0] cannot be modified, the valid range of index is 1--255.

The *pLicense* should not be NULL. The structure pointed by it should contain the new contents for the license module.

Currently, the Activation Code is a 38-byte ANSI string (excluding the terminal **NULL**), therefore, the minimum buffer length should be 38 bytes. If the Activation Code is treated as string, 39-byte buffer will be required, and the terminal **NULL** should be

added at the end of the buffer.

If the *pucActivationCode* is NULL or the buffer length provided by *pulSize* is insufficient, the function will return the error code **SL_INSUFFICIENT_BUFFER** (0x80000006), while the required buffer length will be stored into the variable specified by *pulSize*.

SLGenTimeSyncCode

Generate the Aynchronization Code for adjusting the dongle's internal time. This function is supplied for the software developers to write their own authorization tools.

```
SL_DWORD SLAPI SLGenTimeSyncCode(
   SL_DWORD    ulSLHandle,
   SL_DWORD    ulSerialHi,
   SL_DWORD    ulSerialLo,
   SL_BYTE     *pucTimeSyncCode,
   SL_DWORD    *pulSize
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
ulSerialHi	[IN] The first part of the Serial Number.
ulSerialLo	[IN] The second part of the Serial Number.
pucTimesyncCode	[OUT] Pointer to a buffer to receive the Synchronization Code.
pulSize	[IN] [OUT] Pointer to a integer varialble. As input, provides the length in byte of the buffer pointed by *pucTimesyncCode*. As output, contains the length of the Synchronization Code.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

This function is only applicable for the Master Key.

Currently, the Synchronization Code is a 29-byte ANSI string (excluding the terminal **NULL**), therefore, the minimum buffer length should be 29 bytes. If the Synchronization Code is treated as string, 30-byte buffer will be required, and the terminal **NULL** should be added at the end of the buffer.

If the *pucTimeSyncCode* is NULL or the buffer length provided by *pulSize is insufficient, the function will return the error code **SL_INSUFFICIENT_BUFFER**(0x80000006), while the required buffer length will be stored into the variable specified by *pulSize*.

SLGenMemoryUptPkg

Generate the update package used for the read-only Memory Block in the dongle. This function is supplied for the software developers to write their own authorization tools.

```
SL_DWORD SLAPI SLGenMemoryUptPkg(
   SL_DWORD    ulSLHandle,
   SL_DWORD    ulSerialHi,
   SL_DWORD    ulSerialLo,
   SL_BYTE     *pucData,
   SL_DWORD    ulDataSize,
   SL_BYTE     *pucMemPkg,
   SL_DWORD    *pulSize
);
```

Parameters

ulSLHandle	[IN] Dongle's device handle.
ulSerialHi	[IN] The first part of the Serial Number.
ulSerialLo	[IN] The second part of the Serial Number.
pucData	[IN] Pointer to the buffer containing the data.
ulDataSize	[IN] Data length in byte, currently, it is fixed at 8192.
pucMemPkg	[OUT] Pointer to a buffer to receive the update package.
pulSize	IN] [OUT] Pointer to an integer variable. As input, provides the length in byte of the buffer pointed by *pucMemPkg*. As output, contains the length of the package.

Return Value

SL_SUCCSS(0x80000000) will be returned; otherwise, the error code will be returned.

Remarks

This function is only applicable for the Master Key.

Currently, the length of read-only Memory Block in the dongle is 8192 bytes. Therefore, *ulDataSize* should be 8192. Correspondingly, the length of update package is 9728 bytes. If the *pucMemPkg* is NULL or the buffer length provided by *pulSize is insufficient, the function will return the error code.

SL_INSUFFICIENT_BUFFER(0x80000006), while the required buffer length will be stored into the variable specified by *pulSize*.

API Reference – Error Code

Error Code	Value	Description
SL_SUCCESS	0x80000000	Succeed.
SL_INVALID_PARAMETER	0x80000001	Parameter is invalid.
SL_NO_KEY	0x80000002	No dongle with the specified Developer ID, Product ID or Serial Number is found.
SL_OPEN_FAILED	0x80000003	Fail to open the dongle.
SL_KEY_BUSY	0x80000004	The dongle has been exclusively occupied by other programs
SL_INVALID_HANDLE	0x80000005	Handle is invalid.
SL_INSUFFICIENT_BUFFER	0x80000006	Buffer is insufficient.
SL_ACCESS_DENIED	0x80000007	Access is denied, usually because the password is not verified.
SL_OUT_OF_RANGE	0x80000008	Read or write with invalid data address.
SL_WRONG_PWD	0x80000009	Password is incorrect.
SL_INVALID_LICENSE	0x8000000A	Specified an invalid license index.
SL_LICENSE_NOT_ACTIVE	0x8000000B	License module is not activated.
SL_BAD_UPDATE_PKG	0x8000000C	The contents of Activation Code, update package or Synchronization Code is invalid. Possibly because of reusing the old update data or the serial number in the update data is inconsistent with the dangles' Serial Number.
SL_COMM_ERROR	0x8000000D	Communication error occurs when accessing the dongle.
SL_EEPROM_ERROR	0x8000000E	Error occurs in the EEPROM in the dongle.
SL_LICENSE_EXPIRED	0x8000000F	License module expires.
SL_TIME_ERROR	0x80000010	Dongle's internal time is inconsistent with computer time.
SL_NOT_SET_PID	0x80000011	Product ID needs to be set.
SL_EXCHAGE_KEY_FAILED	0x80000012	Key exchange fails.
SL_PWD_LOCKED	0x80000013	Password is locked. The Password is 16-byte. It allows for 16 times of retry. If the first 8 bytes are correct, but the last 8 bytes are wrong, the frequency of password retry will minus 1. If the first 8 bytes are wrong, the frequency of password retry will remain

Error Code	Value	Description
		unchanged. After the frequency of password retry is reduced to 0, the password will be locked.
SL_PID_SET_COUNT_OVERFLOW	0x80000014	The modification frequency of Product ID exceeds the limit. Product ID can only be modified for 128 times.
SL_CLK_NOT_SUPPORT	0x80000015	Don’t support internal timer.
SL_MEMORY_ERROR	0x80000016	API failed to allocate memory.
SL_SYNC_ERROR	0x80000017	API cannot be synchronized.
SL_VERIFY_DEVICE_FAILED	0x80000018	When opening the dongle, the device verification fails.
SL_ALG_NOT_SUPPORTED	0x80000019	Does not support the algorithm.
SL_INVALID_SEED_SIZE	0x8000001A	When setting the product ID, the length of seed data does not meet the requirements.
SL_NOT_OPENED	0x8000001B	The dongle is not opened or already closed.
SL_DATA_INVALID	0x8000001C	The data in the dongle is corrupt
SL_UNKNOWN_ERROR	0x800000FF	Unknown error.

API Reference – Data Structure

SLDevice

It is a structure for SLEnum function to store the dongle information.

```
typedef struct _SL_DEVICE_
{
   SL_DWORD type;
   SL_DWORD Developer ID;
   SL_DWORD Product ID;
   SL_DWORD serialHi;
   SL_DWORD serialLo;
}SLDevice, *PSLDevice;
```

Members

type	Device type identifier.
Developer ID	Developer ID.
Product ID	Product ID
serialHi	The first part of the Serial Number
serialLo	The second part of the Serial Number

Remarks

The *type* uses flag bit to indicate the dongle type. Its value can take one of the following values:

- **SL_DEVICE_TYPE_NORMAL (0x00000000)**
  End-User Key.
- **SL_DEVICE_TYPE_MASTER (0x00000001)**
  Master Key.
- **SL_DEVICE_TYPE_NETWORK (0x00000004)**
  Network Key.

The above flags can be combined with the following flag:

- **SL_DEVICE_TYPE_CLOCK (0x00000002)**
  Support internal timer.

SL_LICENSE

It is a structure for storing the license module.

```
typedef struct _sl_license_
{
   SL_USHORT flag;
   SL_USHORT count;
   SL_USHORT trialDay;
   SL_DATE    expiredDate;
   SL_USHORT maxNetworkUsage;
} SL_LICENSE, *PSL_LICENSE;
```

Members

flag	Validity identifier.
count	Trial frequency.
trialDay	Trial days.
expirationDate	Authorization expiration date; for date representation methods, please refer to **SL_DATE**. Authorization expiration date uses UTC (Coordinated Universal Time) time.
maxNetworkUsage	Reserved. Not used.

Remarks

The *flag* uses the flag bit to indicate whether the license module is valid, as well as which restriction is used. It value can be combinations of the following values.

- **SL_LIC_FLAG_ACTIVE(0x8000)**
  This license module is avtivated.
- **SL_LIC_FLAG_EXC_COUNT(0x4000)**
  The count filed is valid.
  So the license module is limited to be usable for count times.
- **SL_LIC_FLAG_TRIAL_DAY(0x2000)**
  The trialDay field is valid.
  So the license module is limited to be usable for trialDay days.
- **SL_LIC_FLAG_EXP_DATE(0x1000)**
  The expirationDate field is valid.
  The license module is valid before the specified date.
- **SL_LIC_FLAG_USED(0x0800)**
  This license module has begun to be used. And the dongle has begun to calculate the trialDay.

SL_LIC_FLAG_EXP_DATE and **SL_LIC_FLAG_TRIAL_DAY** should not be used at the same time.

SL_DATE

It is a structure for storing the authorization expiration date.

```
typedef struct _SL_DATE_
{
   SL_USHORT year;
   SL_USHORT month;
   SL_USHORT day;
}SL_DATE, *PSL_DATE;
```

Members

year	Year.
month	Month. Data range: 0—11, 0 = January.
day	Day of month. Data range: 1—31.

SLDeviceInfo

It is a structure for storing the detailed device information.

```
typedef struct
{
   SL_DWORD version;
   SL_DWORD type;
   SL_DWORD Developer ID;
   SL_DWORD Product ID;
   SL_DWORD serialHi;
   SL_DWORD serialLo;
   SL_DWORD algs;
   SL_DWORD maxLicenseNum;
   SL_DWORD maxNetworkUserNum;
   SL_DWORD memBlockNum;
   SLMemAttr attrs[SL_MEM_BLOCK_NUM];
}SLDeviceInfo, *PSLDeviceInfo;
```

Members

version	Version number
type	Device type identifier
Developer ID	Developer ID
Product ID	Product ID
serialHi	The first part of the Serial Number
serialLo	The second part of the Serial Number
algs	Algorithm identifier
maxLicenseNum	The number of license modules
maxNetworkUserNum	The maximum number of network users. Reserved. Not used.
memBlockNum	Number of Memory Blocks
attrs	A list of SLMemAttr structure. Each element contains attributes information for a Memroy Block.

Remarks

The *type* uses flag bit to indicate the dongle type. Its value can take one of the following values:

- **SL_DEVICE_TYPE_NORMAL (0x00000000)**
  End-User Key.
- **SL_DEVICE_TYPE_MASTER (0x00000001)**
  Master Key.
- **SL_DEVICE_TYPE_NETWORK (0x00000004)**
  Network Key.

The above flag bits may be combined with the following flag:

- **SL_DEVICE_TYPE_CLOCK (0x00000002)**
  Support internal timer function.

The *algs* uses flag bit to identify the supported algorithm. Its value can be the combination of the following values:

- **SL_ALG_AES_128 (0x00000001)**
  AES-128.
- **SL_ALG_RSA_1024 (0x00000100)**
  RSA-1024.
- **SL_ALG_SHA1 (0x00020000)**
  SHA1.

The memBlockNum is currently fixed at 2, i.e. two Memroy Blocks.

SLMemAttr

Structure for storing the attributes of the Memory Block in an End-User Key.

```
typedef struct
{
   SL_USHORT size;
   SL_USHORT attribute;
} SLMemAttr;
```

Members

size	Byte count.
attribute	Access attributes.

Remarks

The *attribute* can be the combination of the following values:

- **MEM_ADMIN_READ(0x8000)**
  Readable after verifying the developer password.
- **MEM_ADMIN_WRITE (0x4000)**
  Writable after verifying the developer password.
- **MEM_USER_READ(0x2000)**
  Readable without the need for verifying the developer password.
- **MEM_USER_WRITE(0x1000)**
  Writable without the need for verifying the developer password.
- **MEM_ANONYMOUS_READ(0x0800)**
  Reserved.
- **MEM_ANYOYMOUS_WRITE(0x0400)**
  Reserved.