Elite EL

Remote Update Tools

Developer Guide for Issuer Tool v1.0.0.1

SENSESHIELD

COPYRIGHTS AND TRADEMARKS

The Elite EL with its technical documentation is copyrighted (C) 2017 to present by Senseshield Technology Co., Ltd (Senseshield). 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. Senseshield 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

Company Name: Beijing Senseshield Technology Co., Ltd

Address: Suite 510,Block C,Internet Innovation Center,Building 5,
No.10,Xibeiwang East Road,Haidian District,
Beijing China

Tel.: +86-10-56730936
Fax: +86-10-56730936-8007

Sales: sense@sense.com.cn
Website: www.sense.com.cn

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 EL FAMILY PRODUCTS (including but not limited to the Kit, libraries, utilities, diskettes, disc, Senseshield® and/or Senseshield® keys, the software component of Senseshield and/or EL and the EL License Guide) (hereinafter “Product”) SUPPLIED BY Senseshield Software Technology Co., Ltd (hereinafter "Senseshield") 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 Senseshield®. 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 Senseshield. 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, Senseshield provides 24-month warranty that the Senseshield EL key has no material and manufacturing defects substantially. All the responsibilities of Senseshield 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, Senseshield 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, Senseshield 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 Senseshield EL, the software, the document, other material shipped as accessories, and backups made by you are copyrighted by Senseshield Security GmbH.

- **TERMINATION**

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

CONTENTS

COPYRIGHTS AND TRADEMARKS I

CONTACT II

LICENSE AGREEMENT III

CONTENTS IV

OVERVIEW 5

ABOUT THE GUIDE 5

WHAT ARE EL REMOTE UPDATE TOOLS? 5

ISSUER TOOL 6

WHAT IS ISSUER TOOL? 6

USER ENVIRONMENT 6

DIRECTORIES 6

INTERFACE 7

HOW TO USE 8

DESIGN NEW MODULE 8

EDIT MODULE 10

ADD GENERAL LICENSE 12

ADD MODULE LICENSE 13

DELETE MODULE 13

ADD MODULE TO UPDATING PACKAGE 14

GENERATE UPDATING FILES 15

Set the Package Attributes *15*

Generate Updating Files *16*

QUICK CREATE DATA PACKAGE 16

ISSUER LIB 18

WHAT IS ISSUER LIB? 18

EXPLANATION ON HEADER FILE 18

File Type *20*

Error Code *20*

Property of Data Package *21*

API OF ISSUER LIB 22

e4ru_InitModule *22*

e4ru_AddFile *23*

e4ru_ReleaseModule *25*

e4ru_InitPkg *26*

e4ru_AddModule *28*

e4ru_ProducePkg *29*

e4ru_SProducePkg *30*

e4ru_ReleasePkg *31*

OVERVIEW 1

About the Guide

Mode	Model	Version	Releasing Date
Elite EL Remote Update Tools Client Tool	STD, Genii, RTC, RTCC, NET	v1. 0.0.1	2017.02.22

- **Conventions Used**

The following conventions are used throughout this document:

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 should be careful of.

- **Document Improvement**

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

What are EL Remote Update Tools?

The EL Remote SDK contains three tools: *Initialization Tool*, *Issuer Tool* and *User Update Tool.*

- **Initialization Tool**

InitTool.exe runs at the developer end, mainly support functions to **initialize Issuer Device** (Issuer Key) and **initialize User Device** (User Key), as well as **release information of User Device.**

- **Issuer Tool**

IssuerTool .exe is mainly used by software vendors. This tool is used to **create data module** and **generate update packages.**

- **Client Tool/ User Update Tool**

ClientTool.exe runs at the user end. With this tool, users can **review internal information of device, review information of Update Package** and **update User Device** (User Key)**.**

ISSUER TOOL 2

What is Issuer Tool?

The *IssuerTool .exe* is mainly provided to software vendors. It is capable to perform the following tasks:

- **Create Data Module**
- **Generate Updating Package**

User Environment

OS Supported: Windows NT4.0, Windows 98 2nd, Windows ME, Windows 2000, Windows XP and Windows 2003.

Directories

- Dll: *E4RUIssuer.dll*, *crypt.dll*, *dongle.dll*.
- Language: *language.dll*.
- Issuer Tool: *IssuerTool.exe*.
- Header file corresponding to *E4RUInit.dll: E4RUIssuer.h*.
- Lib file including output symbols of *E4RUIssuer.dll*: *E4RUIsser.lib*.



The operating object of Issuer Tool is the dongle with issuing hardware module, which is firstly recognized and successfully connected by the system.

Interface

Main window of *IssuerTool .exe* is as follows:

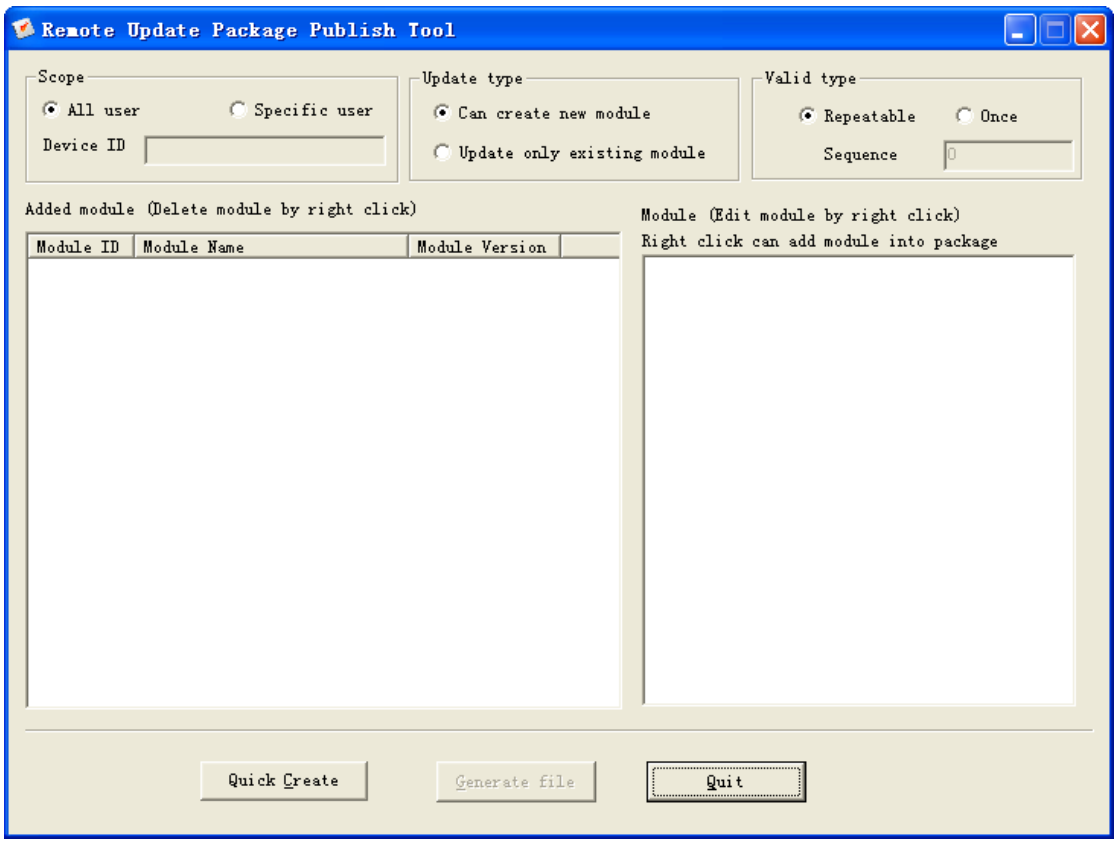


Figure 1

How to Use

3

Design New Module

Right click in right text filed to pop a menu as following:

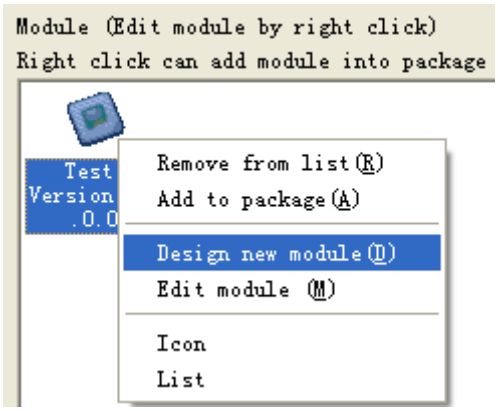


Figure 2

Click **Design new module**, then a Save As dialog box prompts out:

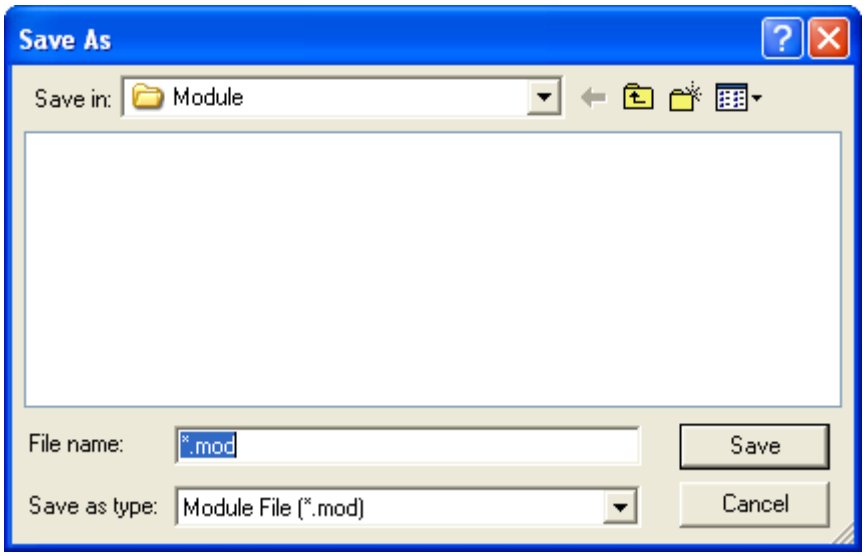


Figure 3

Input the name of file to be saved, and hit the button **Save**, the window **Module Designer** jumps out:

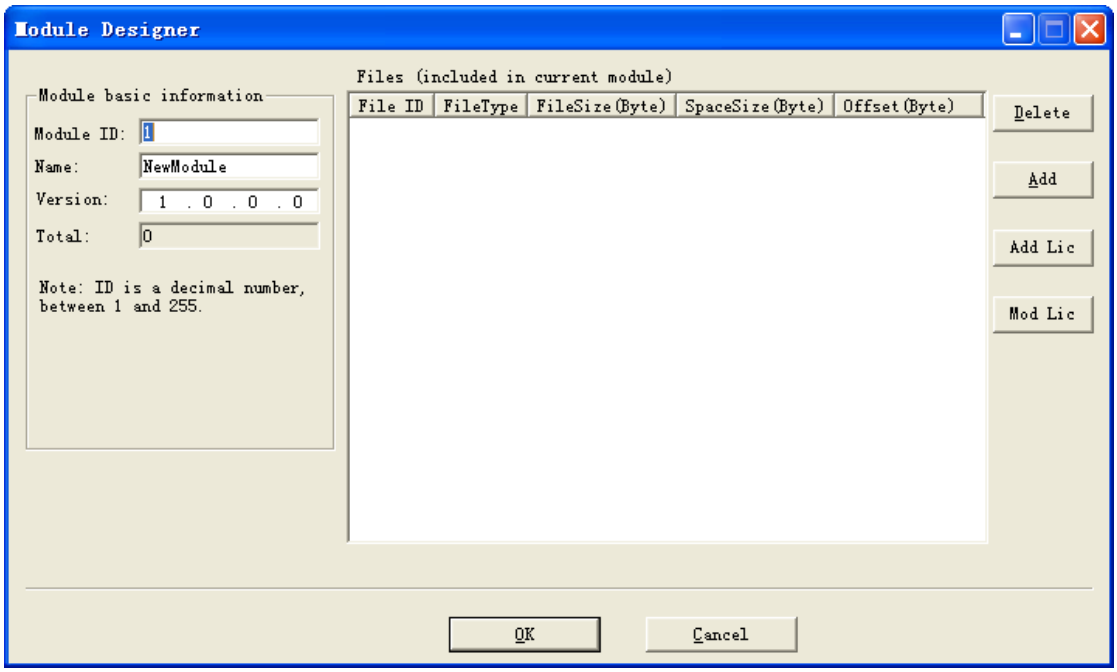


Figure 4

In the **Module Designer**, it is available to modify the **Modules' ID**, **Name**, and **Version** as well as adding files to the module. Click the button **Add**, a dialog box shows up:

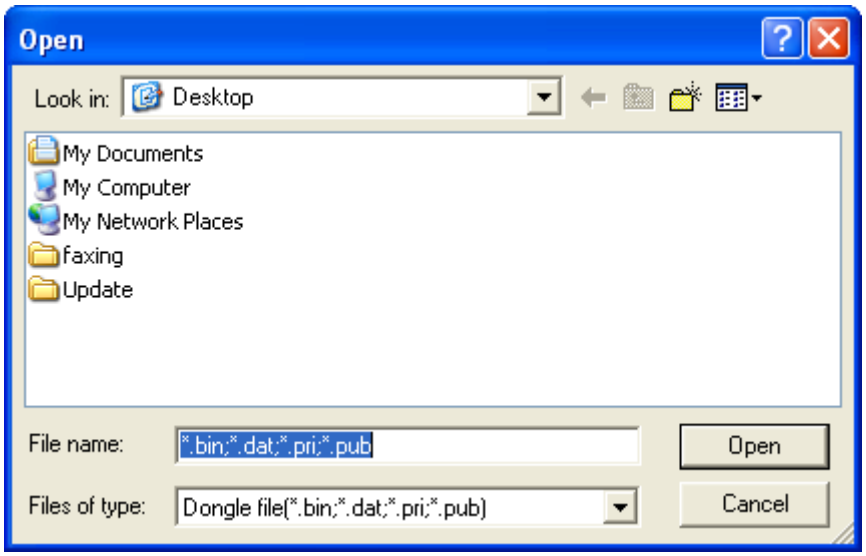


Figure 5

Select a file to be added and hit **Open**. It is available to add multiple files or delete them. After the operation, the result should be as following:

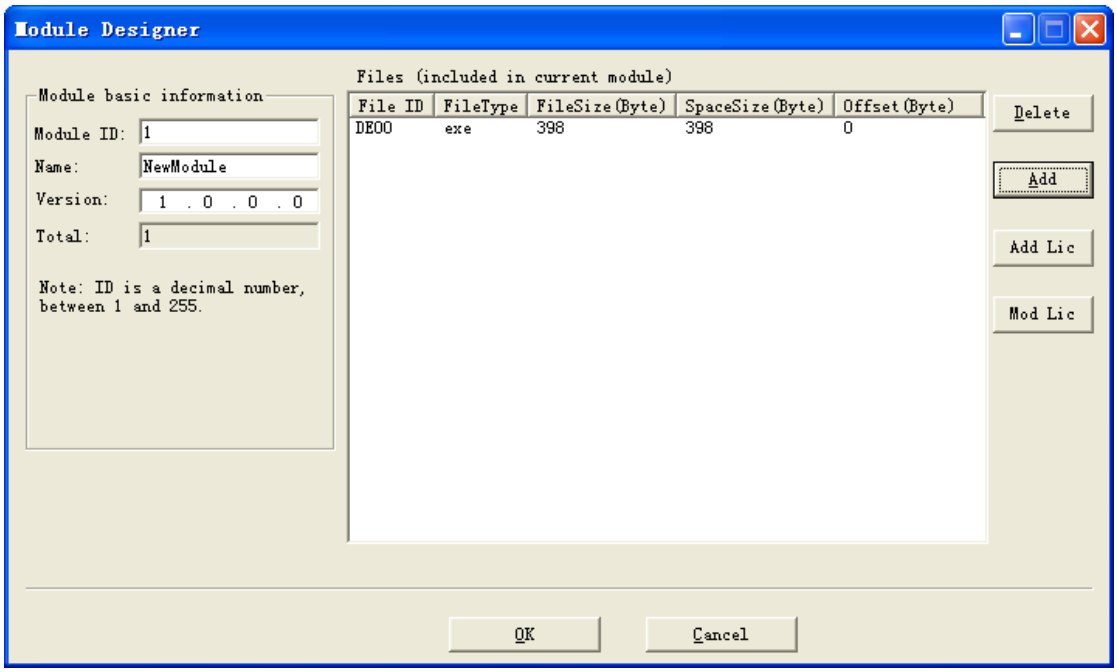


Figure 6

Click OK, and then in the right text field, there is an extra module generated as follows:

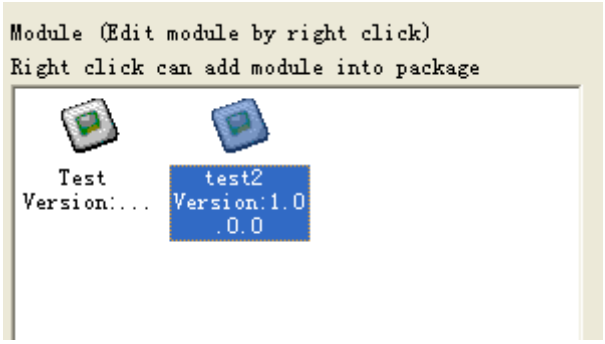


Figure 7

Edit Module

Right click in right text filed to pop a menu and click **Edit module** afterwards as following:

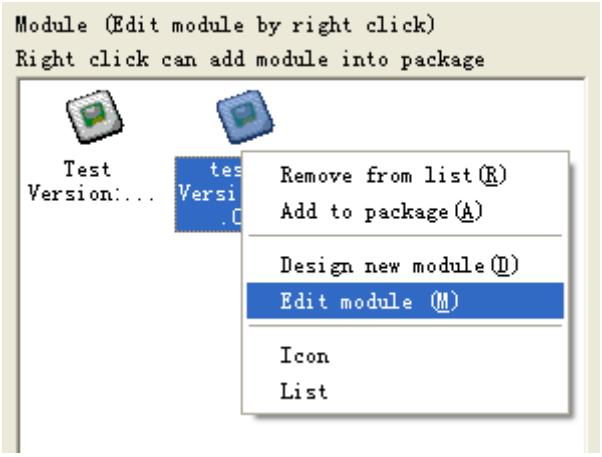


Figure 8

In the **Module Designer**, it is available to modify the **Modules' ID**, **Name**, and **Version** as well as adding or deleting files to the module. For those existing files, it is available to edit their attributes. The steps are: Select the editing file, right click and hit **Property** in the menu popped out.

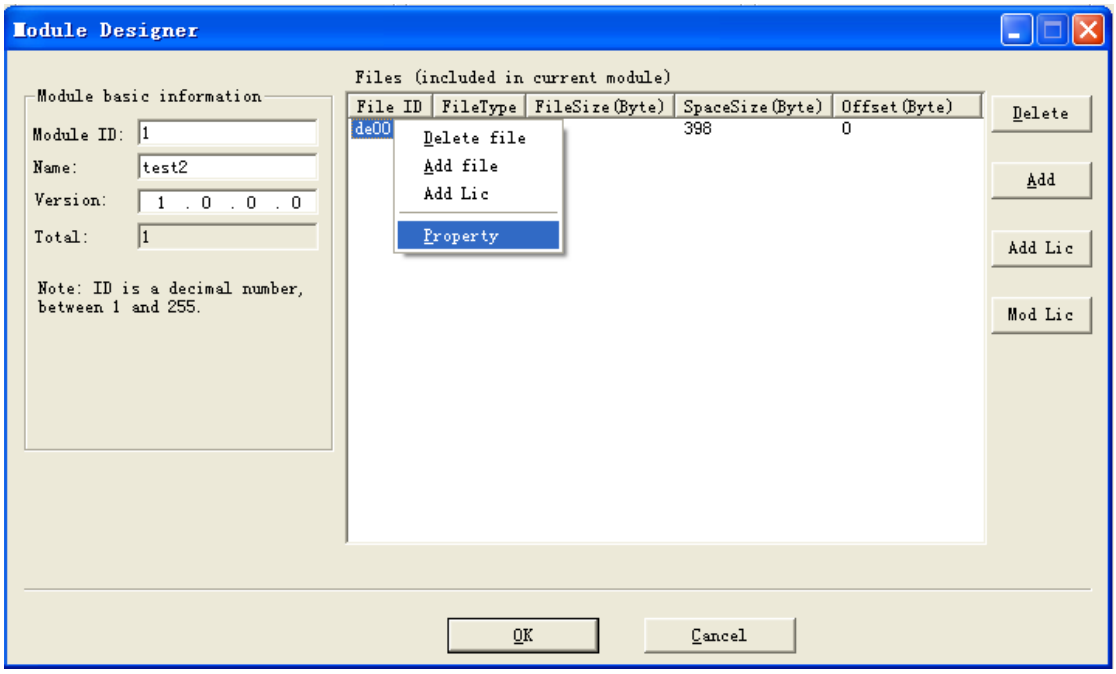


Figure 9

After clicking the **Property**, a dialog box **Modify** will jump out allowing you to change the attributes:

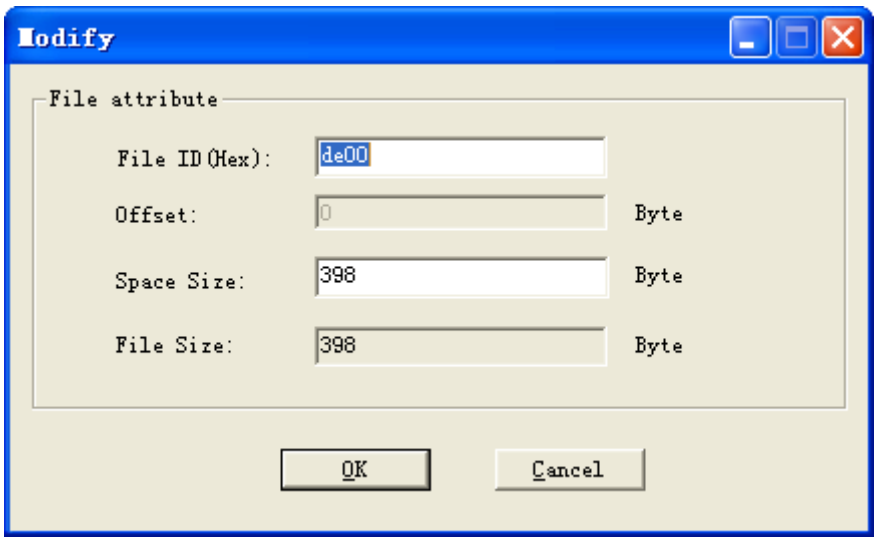


Figure 10

After the changes, click **OK** to go back the **Module Designer** to check out those modifications as follows:

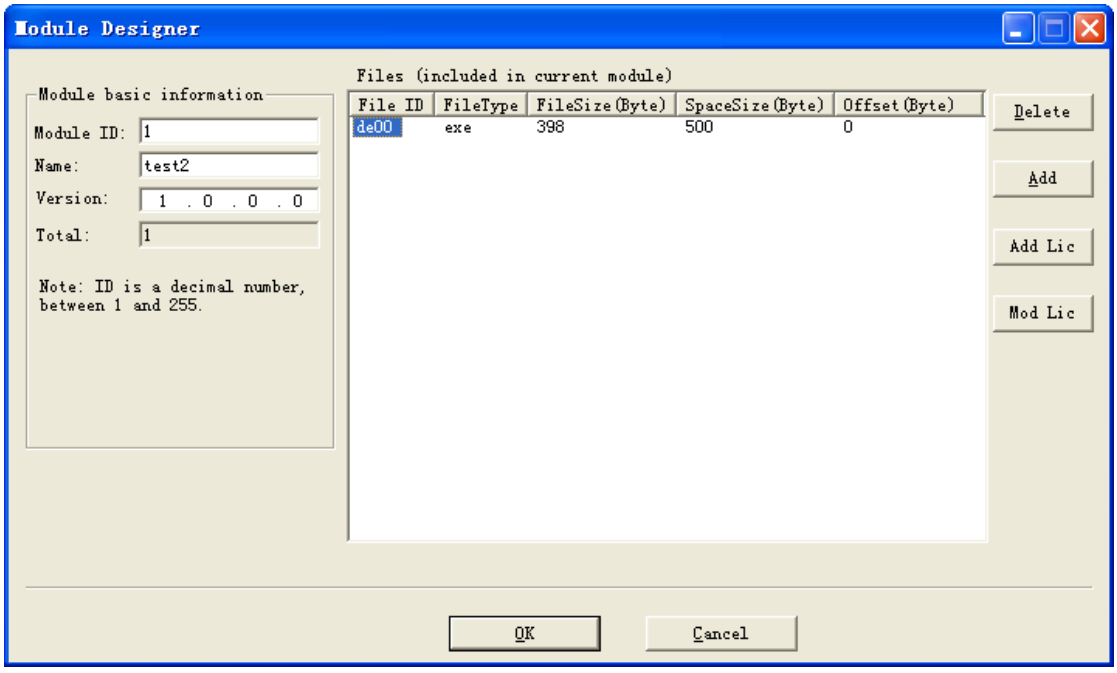


Figure 11

At the moment, click **OK** to save the modification to the current module.

Add General License

In the **Module Designer** to click **Add Lic**, a dialog box **License** will allow you to set the total number of licenses, click **OK** to confirm.

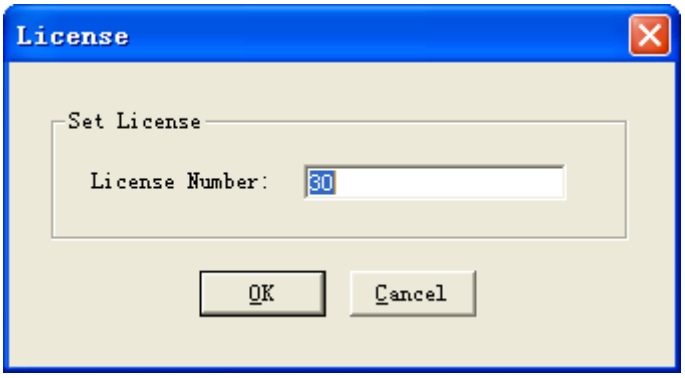


Figure 12

Add Module License

In the **Module Designer**, click **Mod Lic** to get the following:

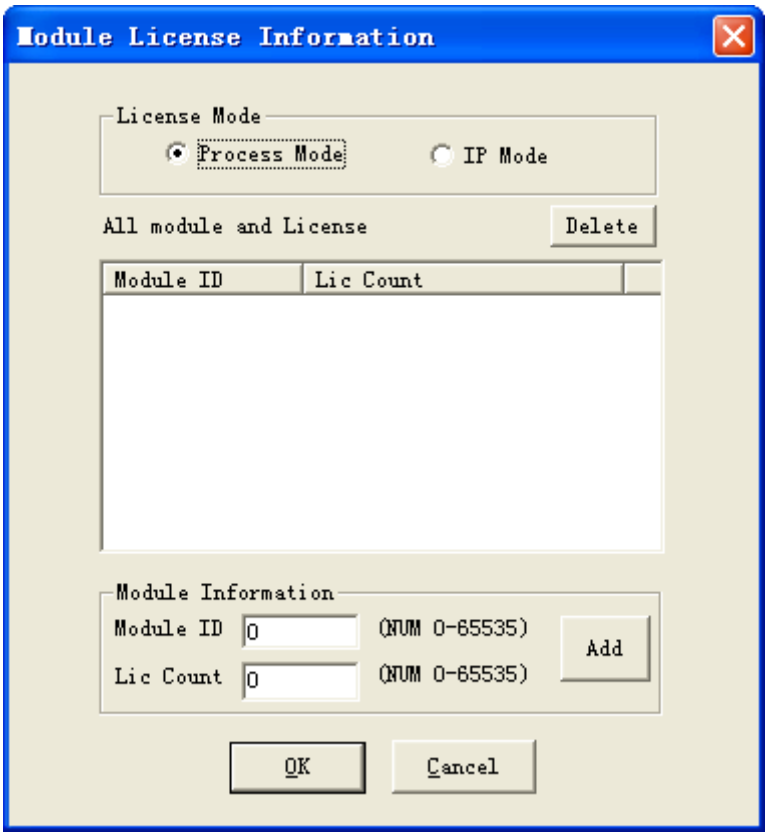


Figure 13

After completion, Click **OK** to confirm.

Delete Module

In the right text filed, select a module to be deleted, and right click, and hit **Remove from List** item from the popped menu as follows:

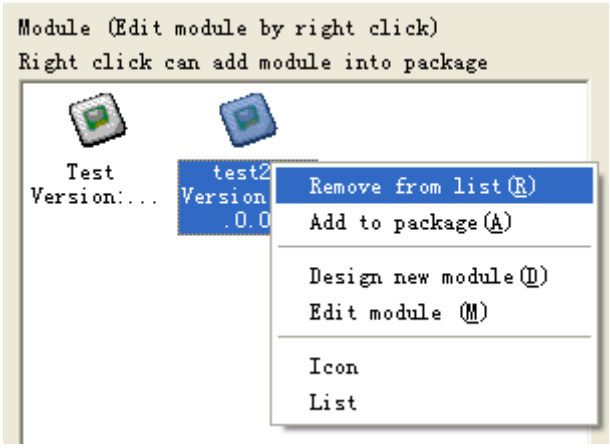


Figure 14

After clicking **OK** in the dialog box to confirm, and then get the following result: the selected module is no longer displayed in the field.

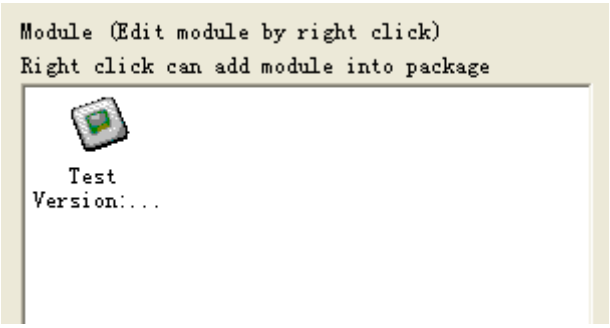


Figure 15

Add Module to Updating Package

Click the module to be added, and right click to hit **Add to package** in the popped menu as following:

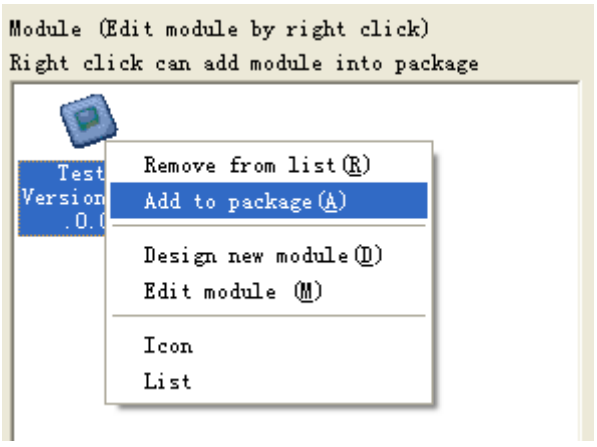


Figure 16

After the completion, the module will be displayed in the list **Added Module**.

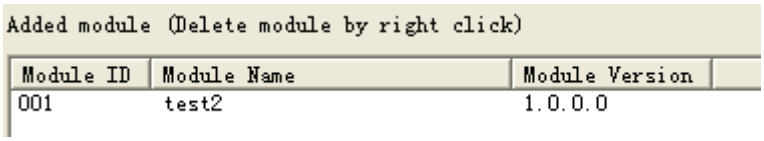


Figure 17

Generate Updating Files

Set the Package Attributes

The upper part of main window is for setting package attributes.

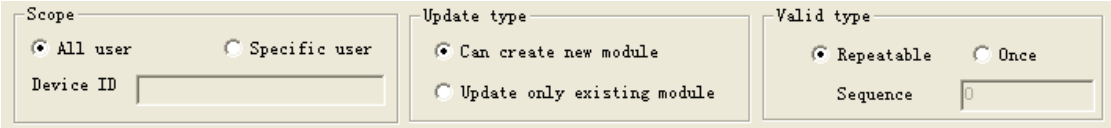


Figure 18

- **User Scope**
  1. When select **All user**, the **Device ID** of User Dongle (Client Device) is not ineffective, all users' device could use this package to update.
  2. When select Specific user, the data package can only update the specified User Dongle, input is Device ID in the field. Only the User Dongle with the specified Device ID can be updatable using the data package.
- **Updating Type**
  1. When select **Can create new module**, no matter the modules in data package exists in the User Dongle or not, it is available to update. If the module is existed, then overwrite the module, if not, create a new module.
  2. When select Update only existing module, and the modules in data package does exist, the content of module can be updated. If not, the update does not work out.
- **Valid Type**
  1. When select Repeatable, and the updating serial number is invalid, the data package in the updating process will compare the updating SN, which could be used to update repeatedly.
  2. When select Once, and the updating SN is valid, in the updating process, the program will compare the updating SN of data package and the one of User Dongle, only in the case of the former is greater than the latter, it is available to update. After the completion, the program will set the updating SN of User Dongle greater than the one of data package; therefore, the data package is no longer usable.

The updating SN of User Dongle has two types:

Global Updating SN and Local Updating SN, when the user scope is chosen **All user**, the

program will compare the Global Updating SN with the updating SN of data package; when the user scope is chosen **Specified User**, the program will compare the Local Updating SN and with the updating SN of data package.

Generate Updating Files

After setting up the attributes, It is necessary to add the needed module to the **Added Module**, Plugging the Issuer Dongle (Issuer Device), then click the button **Generate File** on the main window.

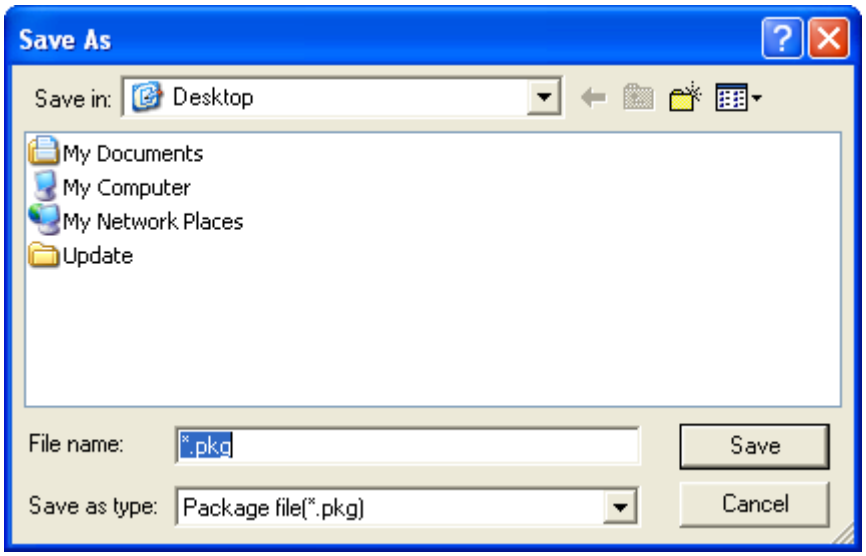


Figure 19

After inputting a file name and hitting **Save**, an updating package is successfully generated by now.

Quick Create Data Package

After setting up the attributes, plug in the Issuer Dongle (Issuer Device), then click the button **Quick Create** on the main window. A dialog box will pop out asking to input a name for data package. Click **Save** and then get the following dialog box.

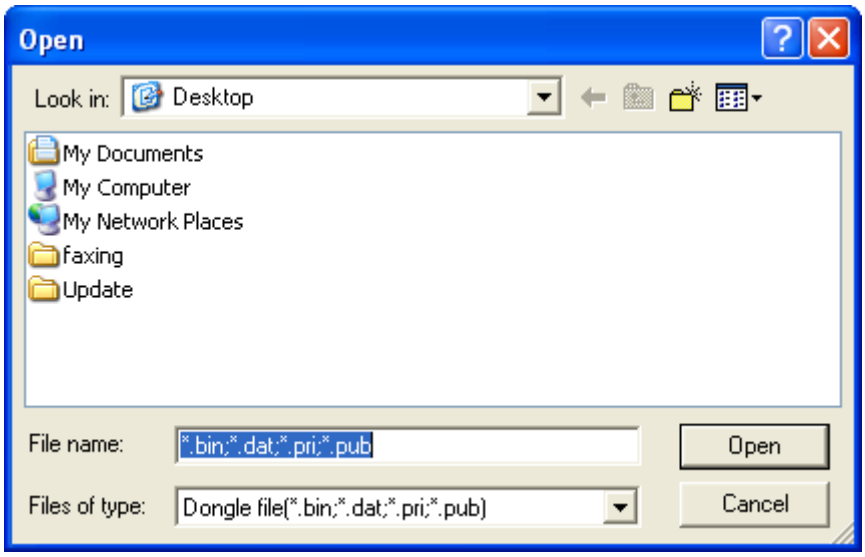


Figure 20

Select the file to be updated and hit **Open**.

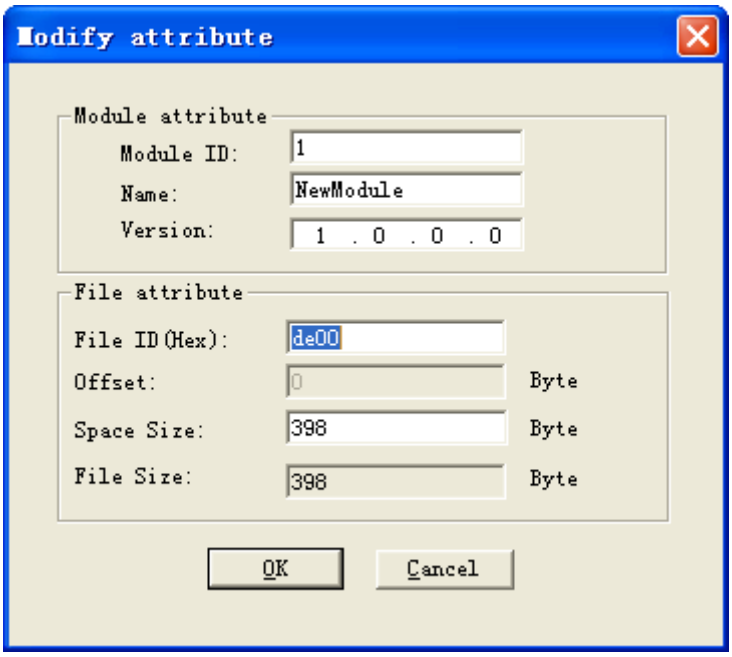


Figure 21

After completion of modifying the attributes of module and file, and click **OK** afterwards. An updating data package is well-generated by now.

ISSUER LIB 4

What is Issuer Lib?

The issuer lib has following functions: *e4ru_InitModule*, *e4ru_AddFile*, *e4ru_ReleaseModule*, *e4ru_InitPkg*, *e4ru_AddModule*, *e4ru_ProducePkg*, *e4ru_ReleasePkg*.

The issuer lib is involved with EF file, module, and package. A module can contain multiple EF files; a package can contain multiple modules. The EF file consists of occupying space, offset original address, content and so on, which all can be added once. The handle is used to represent module and package which have their properties respectively.

In the customization project, it is required to invoke *dongle.dll* in process of invoking initialization lib. You have to use following methods:

1. Using VC compiler, *dongle.dll* and *E4RUIssuer.dll* can be just added into the project.
2. Using other compiler (C++ Builder or Delphi), dongle.dll and E4RUIssuer.dll must be invoked, otherwise calling exception will occur.

Explanation on Header File

The header file *E4RUIssuer.his* as follows:

```
#ifndef __E4RUIssuer_H_INCLUDED__
#define __E4RUIssuer_H_INCLUDED__

#include <windows.h>

#ifndef         IN
#define         IN
#endif

#ifndef         OUT
#define         OUT
#endif

//package's attribute
#define E4RU_FLAG_CREATE_NEW_MODULE       0x01
#define E4RU_FLAG_UPDATE_EXISTING_ONLY    0x02
#define E4RU_FLAG_UPDATE_ONCE             0x04
#define E4RU_FLAG_UPDATE_PERMANENT        0x08
#define E4RU_FLAG_ALL_USER                0x10
#define E4RU_FLAG_INDIVIDUAL_USER         0x12

/* create file type */
#define E4RU_FILE_TYPE_EXE                0x00
#define E4RU_FILE_TYPE_EXE_DATA           0x01
#define E4RU_FILE_TYPE_RSA_PUB            0x02
#define E4RU_FILE_TYPE_RSA_SEC            0x03
#define E4RU_FILE_TYPE_LICENSE            0x04
#define E4RU_FILE_TYPE_MODULE             0x05

//return value definition
#define E4RU_SUCCESS                      0x00000000
#define E4RU_INVALID_PARAMETER            0xE0002002
#define E4RU_INSUFFICIENT_BUFFER          0xE0002003
```

```
#define E4RU_DEVICE_NOT_FOUND              0xE0002004
#define E4RU_FILE_TYPE_MISMATCH            0xE0002005
#define E4RU_MODULE_NO_CONTENT             0xE0002006
#define E4RU_PACKAGE_NO_CONTENT            0xE0002007
#define E4RU_FILE_CREATE_SIZE_ERROR        0xE0002008
#define E4RU_FILE_NAME_RESERVED            0xE0002009
#define E4RU_INVALID_HANDLE                0xE0002201
#define E4RU_GEN_BLOCK_FAIL                0xE0002203
#define E4RU_DIGEST_ERROR                  0xE0002204
#define E4RU_SPACE_NOT_ENOUGH              0xE0002205
#define E4RU_NOT_ISSUER_DEVICE             0xE0002206
#define E4RU_OFFSET_ERROR                  0xE0002207
#define E4RU_SAME_EF_ID_EXIST              0xE0002208
#define E4RU_SAME_MODULE_ID_EXIST          0xE0002209
#define E4RU_EF_FILE_OVERNUMBER            0xE0002210

#ifdef __cplusplus
extern "C" {
#endif

HANDLE WINAPI e4ru_InitModule(
                              IN BYTE Mid,

                              IN DWORD Version,
                              IN LPCSTR lpName
                              );

DWORD WINAPI e4ru_AddFile(
                              IN HANDLE ModuleHandle,
                              IN WORD EfFileID,
                              IN BYTE FileType,
                              IN WORD Size,
                              IN WORD Offset,

                              IN BYTE *Content,
                              IN WORD ContentLen
                              );

DWORD WINAPI e4ru_ReleaseModule(
                              IN HANDLE ModuleHandle

                              );

HANDLE WINAPI e4ru_InitPkg(
                              IN BYTE Scope,
                              IN BYTE ValidType,
                              IN BYTE UpdateType,
                              IN BYTE*pDeviceID,
                              IN DWORD SequenceNum

                              );

DWORD WINAPI e4ru_AddModule(
                              IN HANDLE PkgHandle,
                              IN HANDLE ModuleHandle
                              );

DWORD WINAPI e4ru_ProducePkg(
                              IN PDONGLE4_CONTEXT pS4Ctx,
                              IN BYTE * UserPin,
                              IN HANDLE PkgHandle,
                              OUT BYTE * Buffer,
                              IN OUT DWORD * BufferLen
                              );

DWORD WINAPI e4ru_SProducePkg(
              IN BYTE * TDesKey,
              IN HANDLE PkgHandle,
              OUT BYTE * Buffer,
              IN DWORD * BufferLen
```

```
            );

DWORD WINAPI e4ru_ReleasePkg(
                              IN HANDLE PkgHandle
                              );


#ifdef __cplusplus
       }
#endif
#endif //__E4RUIssuer_H_INCLUDE__
```

You could get the declaration and return values of API by the header file corresponding to *E4RUIssuer.dll*. The following is to explain the details on the return values.

File Type

The type of files added to the module:

Value	Micro Definition	Comment
0x00	E4RU_FILE_TYPE_EXE	Executable Files
0x01	E4RU_FILE_TYPE_EXE_DATA	Data File
0x02	E4RU_FILE_TYPE_RSA_PUB	Public Key File
0x03	E4RU_FILE_TYPE_RSA_SEC	Private Key File
0x04	E4RU_FILE_TYPE_LICENSE	Device Licenses of Network Device
0x05	E4RU_FILE_TYPE_MODULE	Module Licenses of Network Device

Table 1

Error Code

The return values of API:

Value	Micro Definition	Comment
0x00000000	E4RU_SUCCESS	Execution is succeeded
0xE0002002	E4RU_INVALID_PARAMETER	Error of Ineffective parameters
0xE0002003	E4RU_INSUFFICIENT_BUFFER	Insufficient buffer memory
0xE0002004	E4RU_DEVICE_NOT_FOUND	Failed to find device
0xE0002005	E4RU_FILE_TYPE_MISMATCH	Error of EF File type
0xE0002006	E4RU_MODULE_NO_CONTENT	No EF file found from the module
0xE0002007	E4RU_PACKAGE_NO_CONTENT	No module in the updating package
0xE0002008	E4RU_FILE_CREATE_SIZE_ERROR	The space created for EF file is 0
0xE0002009	E4RU_FILE_NAME_RESERVED	The ID of EF file is system reserved.
0xE0002201	E4RU_INVALID_HANDLE	Non-effective input handle
0xE0002203	E4RU_GEN_BLOCK_FAIL	Error of generating updating data block
0xE0002204	E4RU_DIGEST_ERROR	Error of digesting on the updating data package
0xE0002205	E4RU_SPACE_NOT_ENOUGH	The space created for EF file is insufficient
0xE0002206	E4RU_NOT_ISSUER_DEVICE	The connected device is not the Issuer Dongle
0xE0002208	E4RU_SAME_EF_ID_EXIST	A same EF file is existed in the module
0xE0002209	E4RU_SAME_MODULE_ID_EXIST	A same module is existed in the

		package
0xE0002210	E4RU_EF_FILE_OVERNUMBER	The total number of EF file in the module is over 255.

Table 2

Property of Data Package

The property in creating the data package:

Value	Micro Definition	Comment
0x01	E4RU_FLAG_CREATE_NEW_MODULE	To create new module
0x02	E4RU_ FLAG_UPDATE_EXISTING_ONLY	Only update the existing module
0x04	E4RU_ FLAG_UPDATE_ONCE	One-off effective
0x08	E4RU_ FLAG_UPDATE_PERMANENT	Repetitively effective
0x10	E4RU_FLAG_ALL_USER	All Users
0x12	E4RU_FLAG_INDIVIDUAL_USER	Specified User

Table 3

API of Issuer Lib

e4ru_InitModule

Initialize a module

```
HANDLE WINAPI e4ru_InitModule(
             IN  BYTE    Mid,
             IN  DWORD  Version,
             IN  LPCSTR  lpName
             );
```

Parameters:

Mid	[IN] Module ID, cannot be 0.
Version	[IN] Module Version
lpName	[IN] Module Name (<= 16 bytes), the part of length exceeding 16 bytes will be cut off.

Return values:
Returns a handle if succeeded to initialize or NULL

Remarks:
If *lpName* is NULL, then the module name is NULL as well.

e4ru_AddFile

Add the EF file to the module

```
DWORD WINAPI  e4ru_AddFile(
              IN  HANDLE  ModuleHandle,
              IN  WORD    EfFileID,
              IN  BYTE     FileType,
              IN  WORD    Size,
              IN  WORD    Offset,
              IN  BYTE     *Content,
              IN  WORD    ContentLen
              );
```

Parameters:

ModuleHandle	[IN] Module Handle, created by *e4ru_InitModule* or *e4ru_CreateModule*
EfFileID	[IN] The ID of EF file, which must not be the system reserved ID, nor initialized by “F” (For instance, F001). When the EF file type is the device license or module license of network device, this parameter is non-effective.
FileType	[IN] EF File Type, which must be listed in the Table "File Type", or addition will fail.
Size	[IN]EF The space to be occupied by EF File, must not be 0.
Offset	[IN] Offset initial address. It represents where the EF file starts to update and only is applicable on data files, for other types of files, this value must be 0.
Content	[IN]EF The content of EF file
ContentLen	[IN] The length of data included by Content pointer

Return values:

Return E4RU_SUCCESS if succeeded to add EF file, or error code, referring to the Table "Error Code"

Remarks:

- Incoming *ModuleHandle* and *Content* cannot be NULL
- The value of *Size* must be greater than the sum of *Offset* and *ContentLen*.
- The content length included in content pointer must be the same to the value of *ContentLen*.
- The executable file of *FileType* can be only Bin but Hex.
- *Size* cannot be 0.
- If the file ID to be added is same to existing files of the module, then the addition is unable to conduct.
- If the file type is not data file, and *Offset* is not 0, then the addition is unable to conduct.
- If the amount of EF files in a module reaches 255, then it is unable to add new EF

file.

- When the EF file type is the device license of network device, for the max device licenses is 255, the Size and *ContentLen* must be 1, namely 1-byte long.
- When the EF file is the module license of network device, the *Content* must be the declared S4NETCONFIG struct in *dongle.h*.

e4ru_ReleaseModule

Release the module and its content

```
DWORD WINAPI  e4ru_ReleaseModule(
              IN  HANDLE  ModuleHandle
              );
```

Parameters:

ModuleHandle	[IN] Module Handle, created by *e4ru_InitModule* or *e4ru_CreateModule*

Return values:

Return E4RU_SUCCESS if succeeded to add EF file, or error code, referring to the Table "Error Code"

Remarks:

The incoming *ModuleHandle* cannot be NULL.

e4ru_InitPkg

Initialize the updating data package

```
HANDLE  WINAPI  e4ru_InitPkg(
            IN      BYTE      Scope,
            IN      BYTE      ValidType,
            IN      BYTE      UpdateType,
            IN      BYTE      *pDeviceID,
            IN      DWORD   SequenceNum
            );
```

Parameters:

Scope	[IN] User Scope. It must be in the Table "Package Property". E4RU_FLAG_ALL_USER (all user) or E4RU_FLAG_INDIVIDUAL_USER (specified user).
ValidType	[IN] Valid Type. It must be in the Table "Package Property". E4RU_FLAG_UPDATE_ONCE (one-off effective) or E4RU_FLAG_UPDATE_PERMANENT (repetitively effective).
UpdateType	[IN]Update Type. It must be in the Table "Package Property". E4RU_FLAG_CREATE_NEW_MODULE (Create new module) or E4RU_FLAG_UPDATE_EXISTING_ONLY (Update existing module).
pDeviceID	[IN]Device ID, is Global Unique Serial Number in default, 8 bytes. When Scope is E4RU_FLAG_UPDATE_INDIVIDUAL, *pDeviceID* is effective by then. The *pDeviceID* decides on which is the specified User Dongle
SequenceNum	[IN] Global Updating Serial Number (Global Updating Sequence Number). When *ValidType* is E4RU_FLAG_UPDATE_ONCE, *SequenceNum* is effective by then.

Return values:
Returns a handle if succeeded to initialize or NULL

Remarks:

- When pDeviceID is NULL, the returned Handle is NULL.
- When Scope, ValidType, UpdateType is illegal value, then the returned handle is NULL.
- When the value of ValidType is E4RU_FLAG_UPDATE_ONCE, SequenceNum must be greater than or equal to the corresponding sequence number in User Dongle, the data package can be updated in that case.

The incoming *pDeviceID* must be 8 bytes and could be defined as follows:

Byte pDeviceID[8] = {*0x90,0x55,0x66,0x00,0x00,0x00,0x1E,0x23*};

e4ru_AddModule

Add a module to the package

```
DWORD WINAPI  e4ru_AddModule(
              IN  HANDLE  PkgHandle,
              IN  HANDLE  ModuleHandle
              );
```

Parameters:

PkgHandle	[IN] Package Handle, generated by *e4ru_InitPkg*
ModuleHandle	[IN] Module Handle, generated by *e4ru_InitModule* or *e4ru_CreateModule*

Return values:
Return E4RU_SUCCESS if succeeded to add modules or error code, referring to the Table “Error Code”.

Remarks:
The *PkgHandle* and *ModuleHandle* cannot be NULL.

- If the ID of module to be added is same to the existing module in package, it will not available to add.
- If the file of module to be added is existed in the package, it will not available to add the module (Comparing the two file ID are same or not).

e4ru_ProducePkg

Generate an updating package stream according to the content of package.

```
DWORD  WINAPI  e4ru_ProducePkg(
             IN    PDONGLE4_CONTEXT   pS4Ctx,
             IN    BYTE             * UserPin,
             IN    HANDLE             PkgHandle,
             OUT  BYTE              * Buffer,
             IN OUT  DWORD          * BufferLen
             );
```

Parameters:

pS4Ctx	[IN] Device Context
UserPin	[IN] User PIN
PkgHandle	[IN] Package Handle, generated by *e4ru_InitPkg*
Buffer	[OUT] Buffer, storing the generated updating package stream
BufferLen	[IN][OUT] The length of *Buffer*

Return values:
Return E4RU_SUCCESS if succeeded or error code, referring to the Table “Error Code”.

Remarks:
incoming *UserPIN*, *PkgHandle* and *BufferLen* cannot be NULL.

- When incoming *pS4Ctx* is NULL, the function enumerates, opens and closes the device internally.
- When the incoming *Buffer* is NULL, the *BufferLen* will return the length of *buffer* required for the updating package stream calculated and generated by the function.

e4ru_SProducePkg

Generate an updating package stream according to the content of package.

```
DWORD  WINAPI  e4ru_ProducePkg(
            IN       BYTE                * TDesKey,
            IN    HANDLE                 PkgHandle,
            OUT  BYTE                    * Buffer,
            IN OUT  DWORD                * BufferLen
            );
```

Parameters:

TDesKey	[IN] TDES Key
PkgHandle	[IN] Package Handle, generated by *e4ru_InitPkg*
Buffer	[OUT] Buffer, storing the generated updating package stream
BufferLen	[IN][OUT] The length of Buffer

Return values:

Return E4RU_SUCCESS if succeeded or error code, referring to the Table “Error Code”.

Remarks:

The incoming *TDesKey*, *PkgHandle* and *BufferLen* cannot be NULL.

・When the incoming Buffer is NULL, the *BufferLen* will return the length of buffer required for the updating package stream calculated and generated by the function.

e4ru_ReleasePkg

Release the package and its content.

```
DWORD  WINAPI  e4ru_ReleasePkg(
            IN  HANDLE  PkgHandle
            );
```

Parameters:

PkgHandle	[IN] Package Handle, generated by *e4ru_InitPkg*

Return values:

Return E4RU_SUCCESS if succeeded to add modules or error code, referring to the Table “Error Code”.

Remarks:

The incoming *PkgHandle* cannot be NULL.