Elite EL

Remote Update Tools

User Guide for Client 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 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 1

ABOUT THE GUIDE 1
WHAT ARE EL REMOTE UPDATE TOOLS? 1

CLIENT TOOL 2

WHAT IS CLIENT TOOL? 2
USER ENVIRONMENT 2
DIRECTORIES 2
INTERFACE 3

HOW TO USE 5

REVIEW THE DEVICE INFORMATION 5
CHOOSE UPDATING PACKAGE TO INSTALL 6
INSTALL THE UPDATING PACKAGE 6
SAVE STATUS INFORMATION 7
MODIFY DEFAULT USER PIN AND EXECUTION DIRECTORY 8
ERROR INFORMATION 9

USER UPDATE LIB 10

WHAT IS USER UPDATE LIB? 10
HEADER FILE OF USER UPDATE LIB 11
ERROR CODE 13
HEADER STRUCT OF USER DONGLE 14
MODULE INFORMATION STRUCT OF USER DONGLE 15
HEADER STRUCT OF UPDATING PACKAGE 15
MODULE INFORMATION STRUCT OF UPDATING PACKAGE 16

API OF USER UPDATE LIB 17

E4RU_GETDEVICEMODULEINFO 17
E4RU_GETDEVICEMODULEINFO 18
E4RU_GETPKGINFO 19
E4RU_GETPKGMODULEINFO 20
E4RU_UPDATEPKG 21

OVERVIEW 1

About the Guide

Mode	Model	Version	Releasing Date
Elite EL Remote Update Tools Client Tool (User Update Tool)	STD, Genii, RTC, RTCC, NET	v1. 0.0.1	2011.01.29

- **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)**.**

CLIENT TOOL 2

What is Client Tool?

The Client Tool or User Update Tool runs at the client side (user end), providing the following functions:

- **Review the internal information of dongle**
  The information of the User Dongle, after being checked, could be saved in elf file, and opened as text file.
- **Review the information of updating package**
  Before installing updating package, it is available to check its information.
- **Update the User Dongle (Client Device).**
  It is able to update the internal files of dongle by installing the updating package.

User Environment

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

Directories

- Initiation Tool: *InitTool.exe*.
- Dll under directory *bin*: *E4RUClient.dll*, *crypt.dll*, *dongle.dll*.
- Tool: *ClientTool.exe*
- Language: *language.dll*.
- Lib for further development: *E4RUClient.dll*, *E4RUClient.lib*; Static lib: *E4RUClientST.lib*; Corresponding header file:*E4RUClient.h*.



The operating object of Client Tool is the dongle with hardware module of user. The Client Tool will operate on the dongle which is firstly connected to the system successfully.

Interface

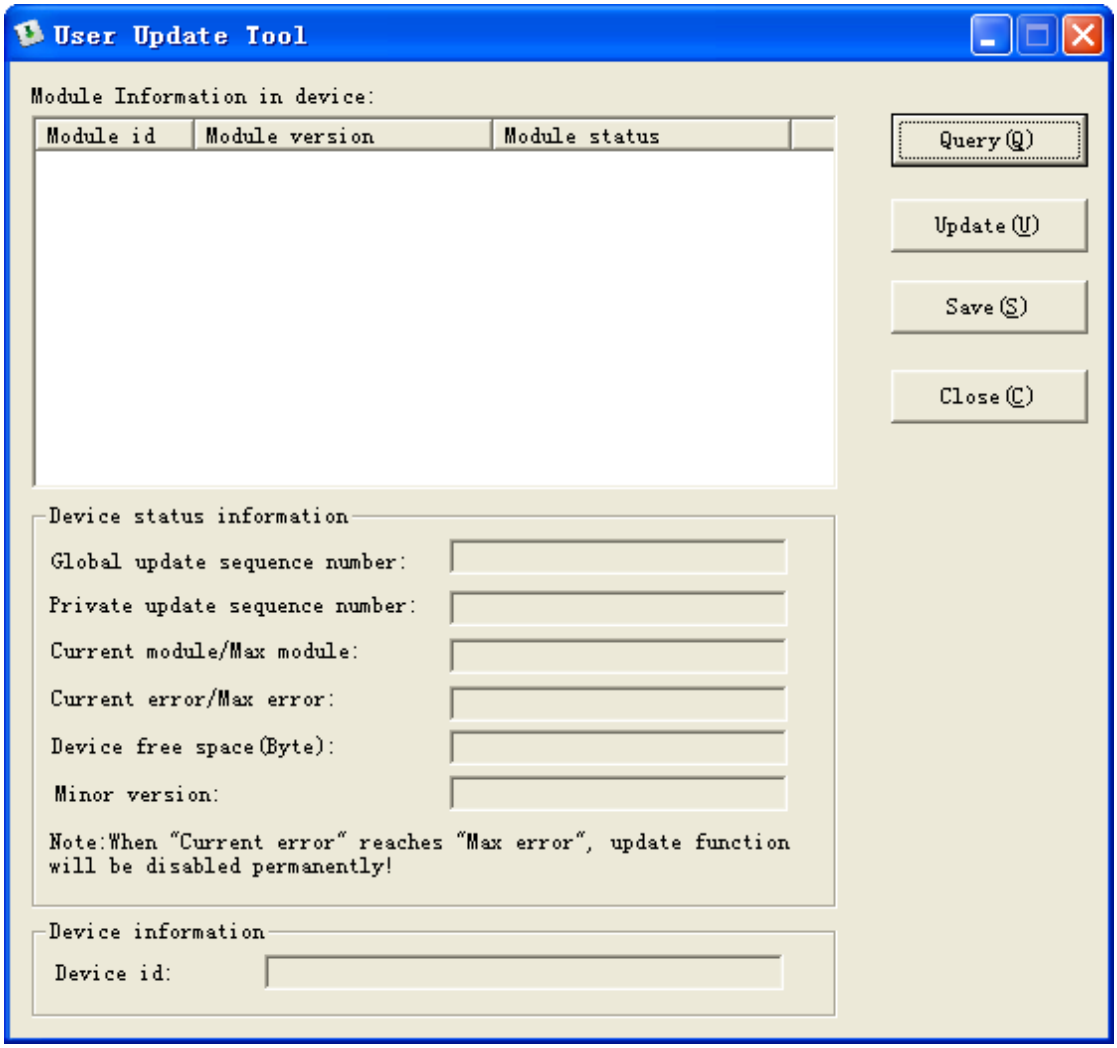


Figure 1

When *ClientTool.exe* runs, the main windows show up. In which, the list of **Module Information in device** will shows up the existing module information; the panel **Device status information** will display the current status of the device. Click the button **Query** to search the information of the User Dongle and display in the current window. Click **Update** to check out the information of updating package, and to install the updating package. Click **Save** to preserve the internal and status information of module in elf file.

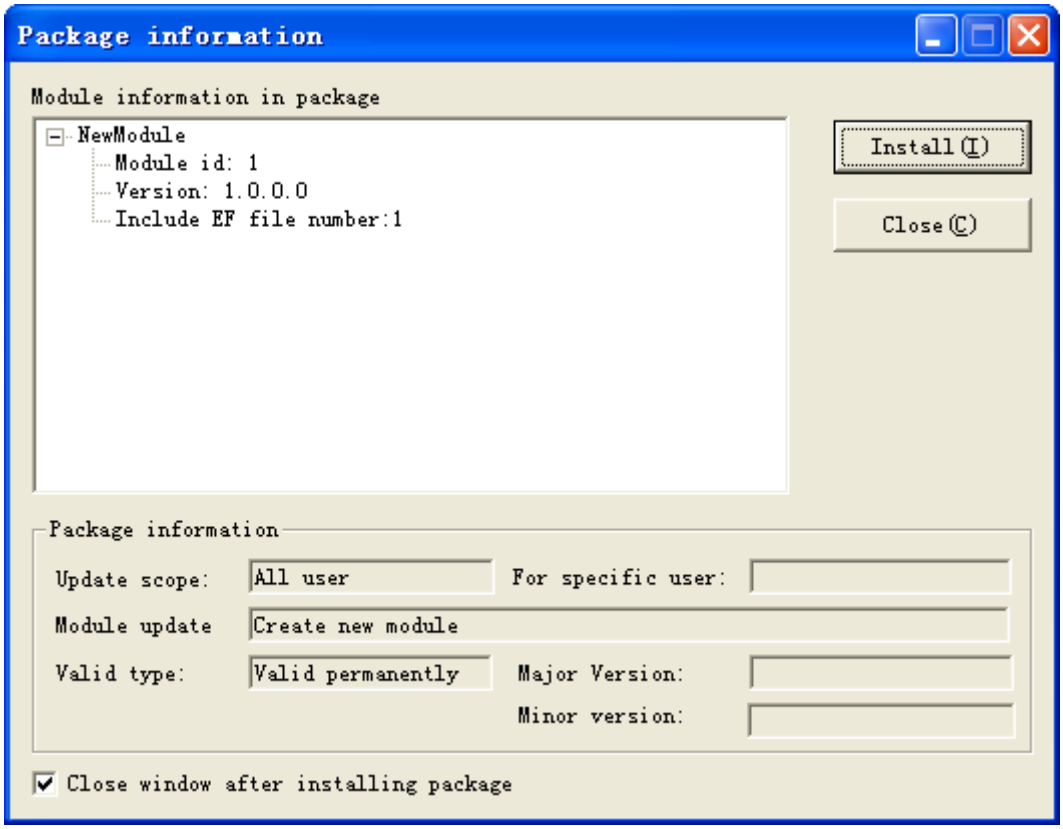


Figure 2

Click **Update** will get the above window. Click **Install** to install the updating package.

How to Use

3

Review the Device Information

Click **Query**, the main window will show the module and status information of current User Dongle as following:

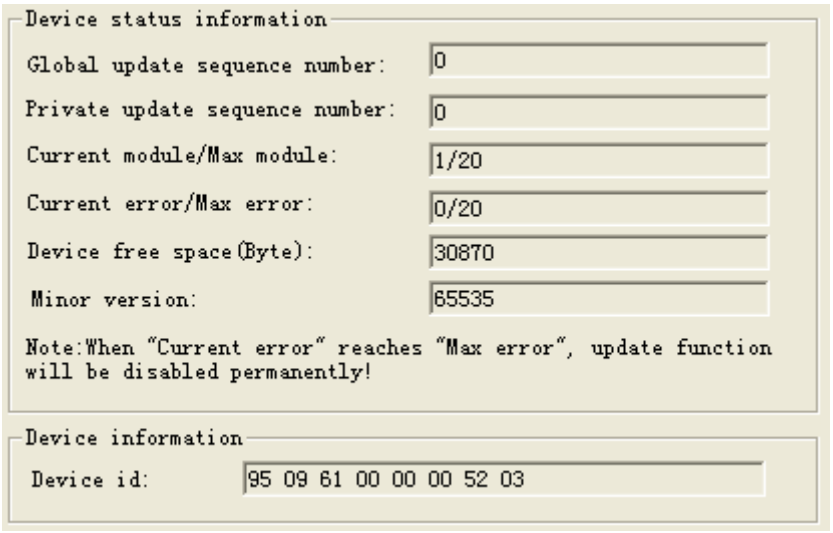


Figure 3

The dongle used in the above sample is a User Dongle just being initialized, therefore no information in panel **Device status Information** (should be Device Module Information), for no module has been set up yet.

Item	**Comment**
Global Updating SN (Global Update Sequence Number)	When the updating package can only be used once to all users, it will check the Global Updating SN
Local Updating SN (Private Updating Sequence Number)	When the updating package can only be used once to specified user, it will check the Local Updating SN
Quantity of Current Modules	
Quantity of Max Modules	The max installable modules of User Dongle
Quantity of Current Errors	Errors of installing updating package
Quantity of Max Errors	The max number that allows the failures of installing updating package, exceeding will seal the updating function.
Remaining Space of Target Directory (Device Free Space)	The remaining space of the directory storing the target file (in byte)
Minor version	When the updating package can be only used once, it is used to show which file is being updated. After the completion, the minor version will be set *0xffff* (65535)
Device ID	When the updating package is aim at the specified user, checking the Device is requested.

Table 1

Choose Updating Package to Install

After click **Update**, and pick up an updating package with suffix *pig*, a dialog box of updating information will pop out as following

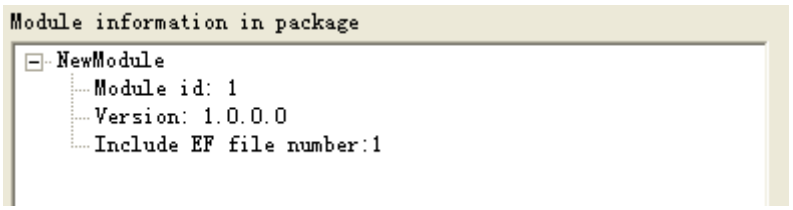


Figure 4

Module Information in Package shows the module of updating package: The module is named **NewModule**, module **version** is 1.0.0.0, and the number of files included is 2.

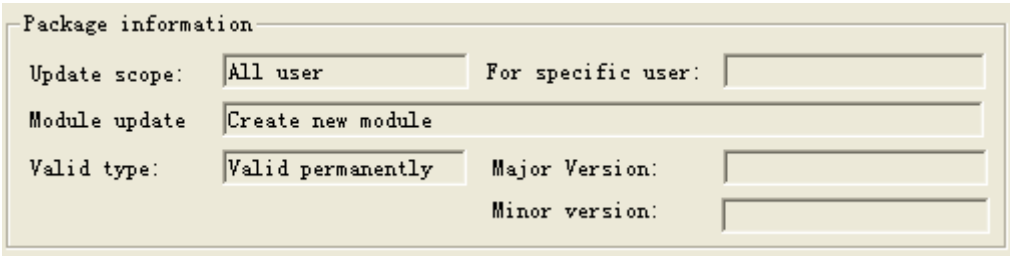


Figure 5

The **Package information** displays the attributes of updating package:

Item	Comment
Update Scope	Display the updating package is for all users or specified user only
For Specific User	When the updating package is for specified user, it should display the Global Unique Serial Number of the User Dongle
Module Updating	When the type is Create New Module, the files of all modules in the updating package will be created or updated in the User Dongle; When the type is Find the module with same name, only the files of valid modules will be created or updated.
Valid Type	Display the updating package is available for once or repetitive installation
Major Version	When the updating package is only for once installation, check the Major version
Minor Version	When the updating package is only for once installation, check the Minor version

Figure 6

Install the Updating Package

Click the button **Install**, the User Dongle starts to setup. After the completion, the result will show up as follows:

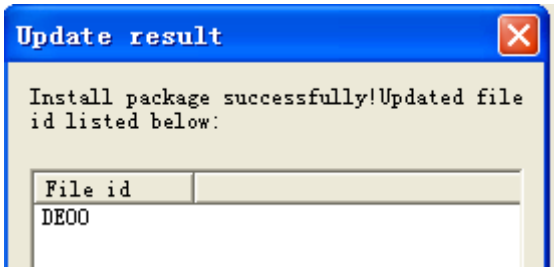


Figure 7

Either in success or failure, the corresponding information will prompt out. After the updating package is successfully installed, go back to the main window, the information of the User Dongle will be displayed as follows:

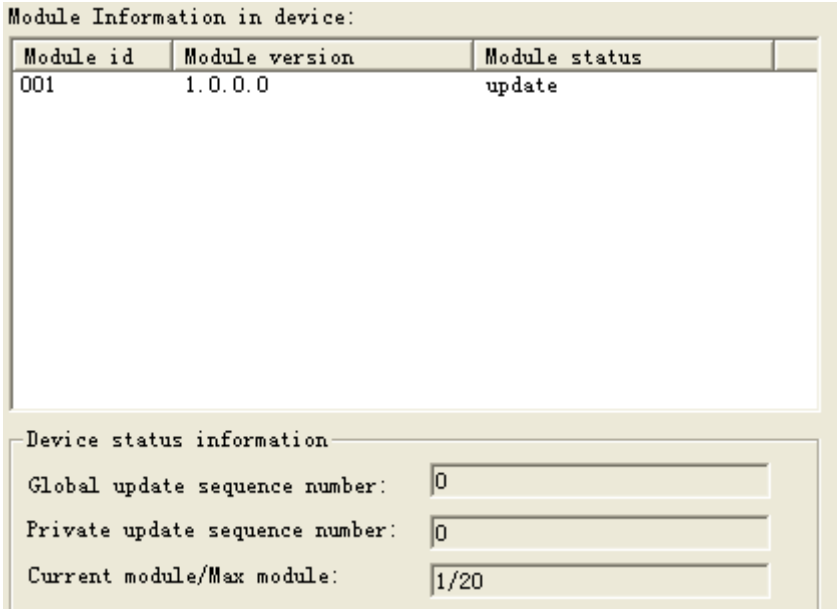


Figure 8

It is clear to see the list **Module Information in device** display the information of installed module, and add 1 to the filed **Current Module**.

If the property of the updating package is the existing module, the tool will only update the existing modules of User Dongle and display neither the **Module ID** which only exists in the updating package nor the User Dongle.

Save Status Information

After the completion of installing updating package, either in success or failure, the tool will create *logFile* directory automatically and save the status file of the User Dongle *date-time.log*. If the user would like to re-save the information file, just click **Save** to store as an *eif* file.

Modify Default User PIN and Execution Directory

The default User PIN and execution directory are stored in *ClientTool.ini* file, and the format is as follows:

```
[ClientTool]
UserPin = 12345678
Dir = \
```

- **"UserPin =" The right side of the equation is the User PIN, which is modifiable.**
- **"Dir =" The right side of the equation is the execution directory, which is modifiable.**
- **As the execution directory is in sub-directory 0001, and the User PIN is "87654321", the *ClientTool.ini* will be amend as follows:**

```
[ClientTool]
UserPin = 87654321
Dir = \0001
```

If "**[ClientTool]**", "**UserPin =**", "**Dir =**" in *ClientTool.ini* are changed or the file itself does not exist, the tool will take the default value for PIN ("12345678") and execution directory ("\\", the root directory). The information is stored in the **recourses** of executable files, which can be edited by VC.

Firstly start VC6, select *ClientTool.exe* and choose Open as Recourses as following:

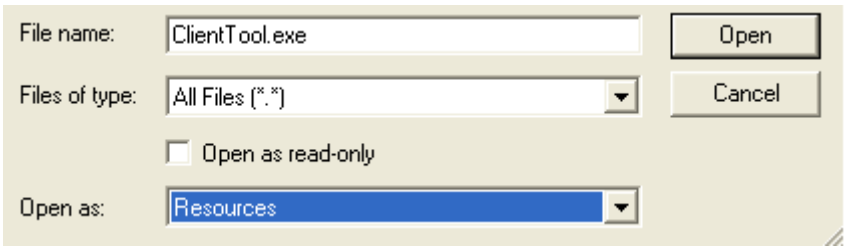


Figure 9

After the opening:

Value	Caption
101	About ClientTool(&A)...
102	12345678
103	\\

Figure 10

It is explicit to see, 102 is corresponding to the User PIN of the User Dongle. Its default value is 12345678 which can be changed to the current User PIN. **103** are corresponding to the directory of hardware module *fb11.bin*. Its default value is "\\". If the hardware module is stored in a sub-directory, then change this value. For instance, \\0001 stands for the hardware module *fb11.bin* storing at the directory \\0001. These values are editable by software vendors.

Error Information

Item	Solution
Fail to open file	The error of opening the updating package file"*.pkg", please check the validity of the file.
Invalid Updating package	The updating package is modified or made by illegal developers
The execution directory of target file does not exist	The default directory is not the one for storing the file of hardware module *fb11*
Invalid User PIN	Failed to verify the User Pin
Failed to obtain the device information	In the process, do not plug in and out the device
The updating failures reaches the max	It means the User Dongle cannot be updated any more.
Insufficient Memory Space	The required space for installing the updating package is greater than the remaining space of directory containing the target file of the User Dongle. It is necessary to request a suitable updating package.
Failed to create a file	Failed to save eif file, please confirm the target file is read-only
The quantity of modules to be created by the updating package exceeds the max module of device	The quantity of modules to be created by the updating package is greater than the addable module quantity (The number of max modules menus the one of current modules) in device. Please request a suitable updating package.
The file in updating package is greater than the one sharing the same name in device	In this case, it is not updatable. Please request a suitable updating package.
The file ID in updating package is invalid (Including the reserved ID)	The IDs of files to be created in device of the updating package includes the reserved ID *0x0000*, *0x0015*, *0x0016*, *0x0018*, *0x001e*, *0x3f00*, *0x3f01*, *0x3f02*, *0x3f03*, *0x3f04* and *0xf**** series file name. Please request a suitable updating package.
The Updating Package is one-off effective	In this case, it is no longer available to install this updating package.
The device is not User Dongle (Client Device)	The connected device is not User Dongle, or the default directory is not the one holding hardware module file *fb11*.
The Updating Package that is not for this User Dongle	The Global Unique Serial Number of specified User Dongle and the one of current User Dongle does not match.
The type of file in updating package and the one sharing same name in device does not match	In this case, please request a suitable updating package.
Please insert a User Dongle	No User Dongle detected.
Failed to install the updating package	The plug-out of device causes the failure of the updating process.
The network service has kicked off, please shut first.	For the service of network dongle has started, it is only available to update remotely after shutting off the service.

USER UPDATE LIB 4

What is User Update Lib?

The user update lib provides functions to obtain the information of the User Dongle, the updating package and to install the updating package. It consists of *E4RUClient.dll* (user update dll), *E4RUInit.lib* (including the output symbol list of *E4RUClient.dll*), *E4RUInit.h* and *dongle.h*.

You could connect *E4UClient.dll* invisibly by *E4RUClient.lib*, or visibly load *E4RUClient.dll*.

The updating function provides a general user interface and at the same time, takes the possibility that user may would like to unify or customize different style into account. So, for issuing you to use the updating function more convenient, *E4RUClient.dll* is supportive to further customization development. This guide is to help you to understand the API that used in *E4RUClient.dll* and accomplish the updating job by few parameters and return values.

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.lib and *E4RUClient.lib* 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.

Header File of User Update Lib

The header file *E4RUClient.h* is as following：

```
#ifndef __E4RU_CLIENT_H__
#define __E4RU_CLIENT_H__

#ifdef __cplusplus
extern "C" {
#endif

#if defined WIN32 || defined _WIN32 || defined _WIN64
#include <windows.h>
#endif

#include “dongle.h”

/*return message definition here*/
#define E4RU_SUCCESS                          0x00000000

#define E4RU_ERROR                            0xE0002001
#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_USERPIN_ERROR                    0xE0002301
#define E4RU_INVALID_EXECUTE_DIR              0xE0002302
#define E4RU_INVALID_PACKAGE                  0xE0002303
#define E4RU_DEVICE_ERROR_MAX                 0xE0002304
#define E4RU_DEVICE_NOT_ENOUGH_SPACE          0xE0002305
#define E4RU_FILE_OVER_THRESHOLD              0xE0002306
#define E4RU_MODULE_MAX                       0xE0002307
#define E4RU_DEVICE_TYPE_ERROR                0xE0002308
#define E4RU_PACKAGE_USED                     0xE0002309
#define E4RU_NEW_MODULE_NOT_INSTALLED         0xE0002310
#define E4RU_INCORRECT_DEVICE_ID              0xE0002311
#define E4RU_GET_DEVICE_TYPE_ERROR            0xE0002312
#define E4RU_GET_LICENSE_FAILED               0xE0002313
#define E4RU_GET_DEVICE_ID_FAILED             0xE0002314
#define E4RU_GET_FREE_SPACE_FAILED            0xE0002315
#define E4RU_CONVERT_BUFFER_FAILED            0xE0002316
#define E4RU_SERVICE_ACTIVE                   0xE0002317
#define E4RU_FREE_LICENSE_FAILED              0xE0002318
#define E4RU_NOT_ENOUGH_MEMORY                0xE0002319
#define E4RU_OPEN_DEVICE_FAILED               0xE0002320
#define E4RU_GET_EF_INFO_ERROR                0xE0002321
#define E4RU_EXECUTE_FAILED                   0xE0002322
#define E4RU_INSTALL_PKG_FAILED               0xE0002323


#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

#pragma pack(push,  1)

typedef struct _DEVICE_MODULE_INFO {
       BYTE    Mid;
```

```
	DWORD	Version;

	BYTE	Valid;
} DEVICE_MODULE_INFO,  *PDEVICE_MODULE_INFO;

typedef struct _SYSTEM_INFOMATION {
	DWORD	GlobalSerial;
	DWORD	PrivateSerial;
	BYTE	MaxError;
	BYTE	ModuleCount;
	BYTE	ErrorNum;
	BYTE	RecordCount;
	WORD	MinorVersion;
	DWORD	FreeSpace;
	BYTE	DeviceId[8];
} SYSTEM_INFORMATION,  *PSYSTEM_INFORMATION;

typedef struct _PACKAGE_INFORMATION {
	BYTE	Scope;
	BYTE	ValidType;
	BYTE	UpdateType;
	BYTE	DeviceID[8];
	DWORD	MajorVersion;
	WORD	MinorVersion;
} PACKAGE_INFORMATION,  *PPACKAGE_INFORMATION;

typedef struct _PACKAGE_MODULE_INFO {
	BYTE	Mid;

	BYTE	Name[16];
	DWORD	Version;

	BYTE	EFNum;
} PACKAGE_MODULE_INFO,  *PPACKAGE_MODULE_INFO;

#pragma pack(pop)

DWORD WINAPI e4ru_GetDeviceInfo(
	IN	DONGLE_CONTEXT	*pS4Ctx,
	IN	BYTE		*pUserPin,
	IN	LPCSTR		lpDirectory,
	OUT  SYSTEM_INFORMATION	*pInfo,
	OUT  DWORD		*pModuleCount
);

DWORD WINAPI e4ru_GetDeviceModuleInfo(
	IN	DONGLE_CONTEXT	*pS4Ctx,
	IN	BYTE		*pUserPin,
	IN	LPCSTR		lpDirectory,
	OUT	DEVICE_MODULE_INFO	*pModuleInfo,
	IN	DWORD		ModuleInfoLen
);

DWORD WINAPI e4ru_GetPkgInfo (
	IN	BYTE		*pPkgContent,
	IN	DWORD		PkgLen,
	OUT	PACKAGE_INFORMATION	*pInfo,
	OUT	DWORD		*pModuleCount
);

DWORD WINAPI e4ru_GetPkgModuleInfo(
	IN	BYTE		*pPkgContent,
	IN	DWORD		PkgLen,

	OUT	PACKAGE_MODULE_INFO	*pModuleInfo,
	IN	DWORD		ModuleInfoLen
);

DWORD WINAPI e4ru_UpdatePkg(
	IN	DONGLE_CONTEXT	*pS4Ctx,
```

```
        IN      BYTE                    *pUserPin,
        IN      LPCSTR                  lpDirectory,
        IN      BYTE                    *pPkgContent,
        IN      DWORD                   PkgLen
);

#ifdef __cplusplus
}
#endif

#endif //__E4RU_CLIENT_H__
```

You could get the declaration and return values of API by the header file corresponding to *E4RUClient.dll*. The following is to explain the details on the return values.

Error Code

Value	Micro Definition	Comment
0x00000000	E4RU_SUCCESS	Execution is succeeded
0xE0002001	E4RU_ERROR	Execution is failed
0xE0002002	E4RU_INVALID_PARAMETER	Ineffective parameters
0xE0002003	E4RU_INSUFFICIENT_BUFFER	Insufficient buffer memory
0xE0002004	E4RU_DEVICE_NOT_FOUND	Failed to find device
0xE0002005	E4RU_FILE_TYPE_MISMATCH	The file type in the updating package does not match the target file in device
0xE0002006	E4RU_MODULE_NO_CONTENT	The module in the updating package does not include the file
0xE0002007	E4RU_PACKAGE_NO_CONTENT	The updating package does not include the module
0xE0002008	E4RU_FILE_CREATE_SIZE_ERROR	The space for creating files in the updating package is 0
0xE0002009	E4RU_FILE_NAME_RESERVED	The file name of the updating package is system reserved.
0xE0002301	E4RU_USERPIN_ERROR	Invalid User PIN
0xE0002302	E4RU_INVALID_EXECUTE_DIR	Error of execution directory of bin. file
0xE0002303	E4RU_INVALID_PACKAGE	Invalid updating package
0xE0002304	E4RU_DEVICE_ERROR_MAX	The updating errors in device has reach the max
0xE0002305	E4RU_DEVICE_NOT_ENOUGH_SPACE	The remaining space of the file directory is insufficient
0xE0002306	E4RU_FILE_OVER_THRESHOLD	The size of the file in updating package is greater than the one of file with same name in device
0xE0002307	E4RU_MODULE_MAX	The amount of modules to be created by the updating package is greater than the remaining creatable number in device
0xE0002308	E4RU_DEVICE_TYPE_ERROR	The device is not User Dongle
0xE0002309	E4RU_PACKAGE_USED	The updating package is used once or its version is lower than the device
0xE0002310	E4RU_NEW_MODULE_NOT_INSTALLED	The update is successful, but does not install the new modules.
0xE0002311	E4RU_INCORRECT_DEVICE_ID	The Device ID specifyied by the

		updating package and the one of User Dongle do not match.
0xE0002312	E4RU_GET_DEVICE_TYPE_ERROR	Failed to obtain the device type
0xE0002313	E4RU_GET_LICENSE_FAILED	Failed to obtain the license
0xE0002314	E4RU_GET_DEVICE_ID_FAILED	Failed to obtain the Device ID
0xE0002315	E4RU_GET_FREE_SPACE_FAILED	Failed to obtain the remaining space of device
0xE0002316	E4RU_CONVERT_BUFFER_FAILED	Failed to convert the data
0xE0002317	E4RU_SERVICE_ACTIVE	The service of network type device has started, Failed to call the function
0xE0002318	E4RU_FREE_LICENSE_FAILED	Failed to release the license
0xE0002319	E4RU_NOT_ENOUGH_MEMORY	Insufficient space
0xE0002320	E4RU_OPEN_DEVICE_FAILED	Failed to open device
0xE0002321	E4RU_GET_EF_INFO_ERROR	Failed to obtain the information of EF file
0xE0002322	E4RU_EXECUTE_FAILED	Failed to execute the file
0xE0002323	E4RU_INSTALL_PKG_FAILED	Failed to install the updating package

Table 2

Header Struct of User Dongle

Its definition is as following:

```
typedef struct _SYSTEM_INFOMATION {
DWORD        GlobalSerial;
DWORD        PrivateSerial;
BYTE         MaxError;
BYTE         ModuleCount;
BYTE         ErrorNum;
BYTE         RecordCount;
WORD         MinorVersion;
DWORD        FreeSpace;
BYTE         DeviceId[8];
} SYSTEM_INFORMATION,  *PSYSTEM_INFORMATION;
```

Member variables of Information header struct in device:

Variable	Type	Scope
GlobalSerial	DWORD	The Global Updating Serial Number (Sequence Number), when the updating package is set one-off effective to all users, it is to check its value. After the updating completion, the value will be the same to the major version of the updating package.
GlobalSerial	DWORD	The Global Updating Serial Number (Sequence Number), when the updating package is set one-off effective to all users, it is to check its value. After the updating completion, the value will be the same to the major version of the updating package.

PrivateSerial	DWORD	The Local (Private) Updating Serial Number (Sequence Number), when the updating package is set one-off effective to specified user, it is to check its value. After the updating completion, the value will be the same to the major version of the updating package.	
MaxError	BYTE	*1~0xff*	Max updating errors allowed.
ModuleCount	BYTE	*1~0xfe*	Max amount of module to be stored in device.
ErrorNum	BYTE	*0~0xff*	The amount of current updating errors
RecordCount	BYTE	*0~0xfe*	The amount of current modules in device
MinorVersion	WORD	Minor Version, records which file is being updated. And when the updating package is one-off effective, it is to check this value, and after the updating completion successfully, the minor version will be set *0xffff* (65535).	
FreeSpace	DWORD	The remaining space of the file directory	
DeviceId[8]	BYTE	Global Unique Serial Number in default, or customized by the developer, to verify the specified user requires to check this value.	

Table 2

Module Information Struct of User Dongle

Its definition is as following:

```
typedef struct _DEVICE_MODULE_INFO {
BYTE         Mid;
DWORD        Version;
BYTE         Valid;
} DEVICE_MODULE_INFO,  *PDEVICE_MODULE_INFO;
```

Member variables of module Information struct in device:

Variable	Type	Value	Comment
Mid	BYTE	1~0xfe	Module ID
Version	DWORD	0~0xffffffff	Module Version
Valid	BYTE	0 or 1	0, if not succeeded 1, if succeeded

Table 3

Header Struct of Updating Package

```
typedef struct _PACKAGE_INFORMATION {
BYTE         Scope;
BYTE         ValidType;
BYTE         UpdateType;
BYTE         DeviceID[8];
DWORD MajorVersion;
WORD         MinorVersion;
} PACKAGE_INFORMATION,  *PPACKAGE_INFORMATION;
```

Member variables of Information header struct in the updating package:

Variable	Type	Value	Comment	Variable

Scope	BYTE	E4RU_FLAG_ALL_USER	Effective to all user	The scope of updating package
		E4RU_FLAG_INDIVIDUAL_USER	Effective to specified user	
ValidType	BYTE	E4RU_FLAG_UPDATE_ONCE	One-off	The using times of updating package
		E4RU_FLAG_UPDATE_PERMANENT	Repeatable	
UpdateType	BYTE	E4RU_FLAG_CREATE_NEW_MODULE	It is available to create new module	The updating module type of updating package
		E4RU_FLAG_UPDATE_EXISTING_ONLY	It is only allowed to update the existing module	
DeviceID[8]	BYTE	Global Unique Serial Number in default, or customized by the developer, to verify the specified user requires to check this value		
MajorVersion	DWORD	Major Version, records the Global Updating Serial Number (Global Updating Sequence Number). When the updating package is one-off effective, it is to check this value.		
MinorVersion	WORD	Minor Version; decrease the amount of files in updating package by -1. And when the updating package is one-off effective, it is to check this value.		

Table 4

• When invoking *e4ru_initPkg*, the incoming parameter *SequenceNum* is the major version of the updating package, namely *SequenceNum* = *MajorVersion*

• To query the information of User Dongle after the update, the major version of updating package corresponds to the Global Updating SN and Local Updating SN. Namely, when the updating package is one-off effective to all user, *MajorVersion* = *GlobalSerial*; when the updating package is one-off effective to specified user, *MajorVersion* = *PrivateSerial*.

Module Information Struct of Updating Package

Its definition is as follows:

```
typedef struct _PACKAGE_MODULE_INFO {
BYTE          Mid;
BYTE          Name[16];
DWORD         Version;
BYTE          EFNum;
} PACKAGE_MODULE_INFO,  *PPACKAGE_MODULE_INFO;
```

Member variables of module Information struct in the updating package:

Variable	Type	Scope	Comment
Mid	BYTE	1~0xfe	Module ID
Name[16]	BYTE		Module Name
Version	DWORD	0~0xffffffff	Module Version
EFNum	BYTE	1~0xff	The amount of files in the module

Table 5

API OF USER UPDATE LIB

e4ru_GetDeviceModuleInfo

Obtain the header information of User Dongle

```
DWORD  WINAPI  e4ru_GetDeviceInfo(
IN           DONGLE_CONTEXT              *pS4Ctx,
IN           BYTE                        *pUserPin,
IN           LPCSTR                      lpDirectory,
OUT          SYSTEM_INFORMATION          *pInfo,
OUT          DWORD                       *pModuleCount);
```

Parameters:

pS4Ctx	[in] Context Pointer to enumerated and opened device
pUserPin	[in] Pointer to User PIN (string) of User Dongle
lpDirectory	[in] Pointer to the file directory (string)
pInfo	[out] Pointer to the information header of device (struct), referring to the Table "Header Struct in Device"
pInfoCount	[in] Pointer to the amount of existing modules in device

Return values:
Return E4RU_SUCCESS, it is available to pass out *pInfo* to obtain the information header of User Dongle and to pass out *pModuleCount* to obtain the amount of existing modules in device, or error code, referring to the Table "Error Code".

Remarks:

- When *pS4Ctx* is NULL, it is able to find the firstly recognized User Dongle
- When *pS4Ctx* is NULL, it is able to open and close the device from internal *E4RUClient.dll*
- If the user opened the device in exclusive mode without closing and NULL will cause the failure of invoking. Therefore, it is recommended to pass in NULL in non-exclusive mode of accessing device, or the opened or enumerated *pS4Ctx*.
- When the device type is network, *lpDirecotry* must be the root directory, namely ("\\").

Requirement:
Hardware Version: Local v2.3.2 or above, Network v2.0.5 or above

e4ru_GetDeviceModuleInfo

Obtain the module information of User Dongle

```
DWORD  WINAPI  e4ru_GetDeviceModuleInfo(
IN          DONGLE_CONTEXT              *pS4Ctx,
IN          BYTE                        *pUserPin,
IN          LPCSTR                      lpDirectory,
OUT         DEVICE_MODULE_INFO          *pModuleInfo,
IN          DWORD                       ModuleInfoLen);
```

Parameters:

pS4Ctx	[in] Context Pointer to enumerated and opened device
pUserPin	[in] Pointer to User PIN (string) of User Dongle
lpDirectory	[in] Pointer to the file directory (string)
pModuleInfo	[out] Pointer to the module information in device (struct), referring to the Table “Module Struct in Device”
ModuleInfoLen	[in] The size of module information

Return values:
Return E4RU_SUCCESS, it is available to pass out *pModuleInfo* to obtain the module information of User Donge, or error code, referring to the Table “Error Code”.

Remarks:

- When *pS4Ctx* is NULL, it is able to find the firstly recognized User Dongle
- When *pS4Ctx* is NULL, it is able to open and close the device from internal *E4RUClient.dll*
- If the user opened the device in exclusive mode without closing and NULL will cause the failure of invoking. Therefore, it is recommended to pass in NULL in non-exclusive mode of accessing device, or the opened or enumerated *pS4Ctx*.
- When no modules exists in device (*ModuleInfoLen* = 0), it is not neccesory to call this function, or E4RU_INVALID_PARAMETER returns.
- When the device type is network, *lpDirecotry* must be the root directory, namely (“\\”).

Requirement:
Hardware Version: Local v2.3.2 or above, Network v2.0.5 or above

Sample:
By calling *e4ru_GetDeviceInfo* to get the amount of existing modules in device *pModuleCount*, to take *ModuleInfoLen* = *pModuleCount* * *sizeof*(DEVICE_MODULE_INFO) to pass in as the size of module information.

e4ru_GetPkgInfo

Obtain the header information of the updating package

```
DWORD  WINAPI  e4ru_GetPkgInfo(
IN           BYTE                  *pPkgContent,
IN           DWORD                 PkgLen,
OUT          PACKAGE_INFORMATION   *pInfo,
OUT          DWORD                 *pModuleCount);
```

Parameters:

pPkgContent	[in] Pointer to the content of updating package (byte stream)
PkgLen	[in] Length of the content of updating package (byte stream)
pInfo	[out] Pointer to the header information of updating package (struct) , referring to the Table “Header Struct in Package”
pModuleCount	[out] Pointer to the amount of modules of updating package

Return values:

Return E4RU_SUCCESS, it is available to pass out *pInfo* to obtain the information header of User Dongle and to pass out *pModuleCount* to obtain the amount of existing modules in device, or error code, referring to the Table “Error Code”.

e4ru_GetPkgModuleInfo

Obtain the module information of the updating package

```
DWORD  WINAPI  e4ru_GetPkgModuleInfo(
IN           BYTE                  *pPkgContent,
IN           DWORD                 PkgLen,
OUT          PACKAGE_MODULE_INFO   *pModuleInfo,
IN           DWORD                 ModuleInfoLen);
```

Parameters:

pPkgContent	[in] Pointer to the content of updating package (Byte Stream)
PkgLen	[in] Pointer to User PIN (string) of the updating package
pModuleInfo	[out] Pointer to the module information in the updating package (struct), referring to the Table "Module Struct in Package"
ModuleInfoLen	[in] The size of module information of the updating package

Return values:
Return E4RU_SUCCESS, it is available to pass out *pModuleInfo* to obtain the module information of package, or error code, referring to the Table "Error Code".

Sample:
By calling *e4ru_GetPackageInfo* to get the amount of existing modules in device *pModuleCount*, to take *ModuleInfoLen* = *pModuleCount* * *sizeof*(PACKAGE_MODULE_INFO) to pass in as the size of module information.

e4ru_UpdatePkg

Install the updating package

```
DWORD  WINAPI  e4ru_UpdatePkg(
IN    DONGLE_CONTEXT        *pS4Ctx,
IN    BYTE                  *pUserPin,
IN    LPCSTR                lpDirectory,
IN    BYTE                  *pPkgContent,
IN    DWORD                 PkgLen);
```

Parameters:

pS4Ctx	[in] Context Pointer to enumerated and opened device
pUserPin	[in] Pointer to User PIN (string) of User Dongle
lpDirectory	[in] Pointer to the file directory (string)
pPkgContent	[in] Pointer to the content of updating package (byte stream)
PkgLen	[in] Length of the content of updating package

Return values:

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

The updating process will fail in the following cases:

- Use the one-off updating package which was used, or the version of updating package is lower than the device.
- The updating package is not specified for the User Dongle.
- The type of file in the updating package and the one with same name do not match.
- The space of file directory in device is insufficient
- The file name are system reserved, including *0x0000, 0x0015, 0x0016, 0x0018, 0x001e, 0x3f00, 0x3f01, 0x3f02, 0x3f03, 0x3f04 and 0xf**** series.
- The updating requires to create new module, but the amount of modules in device has reached the max.
- The amount of updating errors in device has reached the max.
- When updating the files in device, the size of files in the updating package is greater than the one of file with same name in device.
- Invalid User PIN
- The file directory does not exist in device.
- The updating package is being modified.

Remarks:

- When *pS4Ctx* is NULL, it is able to find the firstly recognized User Dongle
- When *pS4Ctx* is NULL, it is able to open and close the device from internal *E4RUClient.dll*
- If the user opened the device in exclusive mode without closing and NULL will cause the failure of invoking. Therefore, it is recommended to pass in NULL in non-exclusive mode

of accessing device, or the opened or enumerated *pS4Ctx*.

• When the device type is network, *lpDirecotry* must be the root directory, namely ("\\").

Requirement:
Hardware Version: Local v2.3.2 or above, Network v2.0.5 or above