

# **XPOS**

## **Secondary development interface document**

## Document management

### Version history

Date	Version	Modify record	Author
20181109	1.0	Basic interface	yangjy

# Content

<b>Content</b> .....	<b>3</b>
<b>1 profile</b> .....	<b>9</b>
1.1 overall structure.....	9
1.1 modular design .....	9
<b>2 System module(libapi_system)</b> .....	<b>10</b>
2.1 interface list .....	10
2.2 API interface.....	11
2.2.1 System initialization(Sys_Init) .....	11
2.2.2 Vendor personality parameter setting(Sys_Config).....	12
2.2.3 Get terminal info(Sys_GetTerminalInfo) .....	13
2.2.4 Get system time(Sys_GetDateTime) .....	13
2.2.5 Set systemn time(Sys_SetDateTime).....	14
2.2.6 Scan buttun(Sys_CheckKey).....	15
2.2.7 Clear button cache(Sys_ClrKey).....	15
2.2.8 Open timer(Sys_TimerOpen) .....	16
2.2.9 Test timer (Sys_TimerCheck) .....	16
2.2.10 Turn of timer(Sys_TimerClose) .....	17
2.2.11 delay(Sys_Delay).....	18
2.2.12 Terminal sleep (Sys_Sleep) .....	18
2.2.13 Terminal reboot(Sys_Reboot).....	19
<b>3 Tool module (libapi_util)</b> .....	<b>19</b>
3.1 interface list .....	19
3.2 API interface.....	21
3.2.1 ASCII code change to BCD code (Util_Asc2Bcd) .....	21
3.2.2 BCD code convert to ASCII code (Util_Bcd2Asc).....	21
3.2.3 Int type data convert to BCD code (Util_Int2Bcd).....	22

3.2.4	BCD code convert to int type(Util_Bcd2Int) .....	23
3.2.5	Caculate LRC(Util_GenLrc).....	24
3.2.6	DES encryption and decryption (Util_Des) .....	24
3.2.7	Chinese character copy(Util_StrCopy).....	25
3.2.8	Waiting button(Util_WaitKey) .....	26
3.2.9	Input method input(Util_InputMethod).....	27
3.2.10	String input (Util_InputText) .....	28
3.2.11	Amount input(Util_InputAmount).....	29
3.2.12	IP input (Util_InputIp).....	30
3.2.13	beep(Util_Beep).....	31
3.2.14	Voice play (Play_Voice) .....	32
3.2.15	Generate random numbers(Util_Rand) .....	33
<b>4</b>	<b>File module(libapi_file) .....</b>	<b>34</b>
4.1	Interface list .....	34
4.2	API interface.....	35
4.2.1	Check if the file exists (UFile_Check) .....	35
4.2.2	File open / create(UFile_OpenCreate).....	36
4.2.3	File read(UFile_Read) .....	37
4.2.4	Write file (UFile_Write) .....	38
4.2.5	Positioning file pointer(UFile_Lseek).....	39
4.2.6	Delete file record (UFile_Delete) .....	40
4.2.7	Close file (UFile_Close) .....	41
4.2.8	Delete file (UFile_Remove) .....	42
4.2.9	Rename file (UFile_Rename).....	42
4.2.10	Empty file(UFile_Clear).....	43
4.2.11	Get free space(UFile_GetFreeSpace).....	44
4.2.12	Get the number of file records (UFile_GetNumberOfRecords) .....	45
4.2.13	Append file record(UFile_AppendRecord).....	46

4.2.14	Query records based on index number(UFile_GetRecordByIndex).....	47
4.2.15	Check record(UFile_GetRecord) .....	48
4.2.16	Update record(UFile_UpdateRecord).....	49
4.2.17	Update records based on index number(UFile_UpdateRecordByIndex) .....	51
4.2.18	Delete record (UFile_DeleteRecord) .....	52
4.2.19	Delete records based on index number(UFile_DeleteRecordByIndex) .....	53
4.2.20	Read one line text(UFile_ReadLine) .....	54
4.2.21	Read non-fixed length records (UFile_ReadTLV).....	55
4.2.22	Write non-fixed data(UFile_WriteTLV) .....	56
4.2.23	Delete non-fixed length record (UFile_DeleteTLV).....	57
<b>5</b>	<b>IC card module (libapi_iccard) .....</b>	<b>58</b>
5.1	interface list .....	58
5.2	API interface.....	59
5.2.1	Turn on IC card device (Icc_Open).....	59
5.2.2	Turn off IC card device (Icc_Close).....	60
5.2.3	Turn off IC card device (Icc_Close).....	60
5.2.4	Test card(Icc_GetCardStatus).....	62
5.2.5	Contact card power on(Icc_PowerUp) .....	62
5.2.6	Contact card power off (Icc_PowerDown) .....	63
5.2.7	Contact card communication (Icc_ICComm).....	64
5.2.8	NFC card searching card (Icc_CTLSPowerUpAndSeek) .....	66
5.2.9	NFC card power off(Icc_CTLSPowerDown).....	67
5.2.10	Use APDU to communicate with NFC card(Icc_CTLSComm).....	67
<b>6</b>	<b>communication ( libapi_comm) .....</b>	<b>69</b>
6.1	interface list .....	69
6.2	API interface.....	69
6.2.1	comm_net_link.....	69
6.2.2	comm_net_unlink.....	70

6.2.3	<i>comm_sock_connect</i> .....	71
6.2.4	<i>comm_sock_recv</i> .....	71
6.2.5	<i>comm_sock_send</i> .....	72
6.2.6	<i>comm_sock_close</i> .....	73
6.2.7	<i>comm_ssl_init</i> .....	73
6.2.8	<i>comm_ssl_connect</i> .....	74
6.2.9	<i>comm_ssl_send</i> .....	75
6.2.10	<i>comm_ssl_recv</i> .....	75
6.2.11	<i>comm_ssl_close</i> .....	76
<b>7</b>	<b>security ( libapi_security )</b> .....	<b>77</b>
7.1	interface list .....	80
7.2	API interface .....	80
7.2.1	<i>mksk_save_plaintext_key</i> .....	80
7.2.2	<i>mksk_save_encrypted_key</i> .....	81
7.2.3	<i>mksk_3des_run</i> .....	82
7.2.4	<i>dukpt_get_ksn</i> .....	83
7.2.5	<i>dukpt_3des_run</i> .....	83
7.2.6	<i>dukpt_init_key</i> .....	84
<b>8</b>	<b>Gui (libapi_gui)</b> .....	<b>86</b>
8.1	interface list .....	86
8.2	API interface .....	87
8.2.1	<i>gui_bar_rc</i> .....	87
8.2.2	<i>gui_set_bar_color</i> .....	88
8.2.3	<i>gui_get_bar_color</i> .....	89
8.2.4	<i>gui_set_font</i> .....	89
8.2.5	<i>gui_get_font</i> .....	90
8.2.6	<i>gui_set_text_color</i> .....	91
8.2.7	<i>gui_get_text_color</i> .....	91

8.2.8	<i>gui_set_text_bg_color</i> .....	92
8.2.9	<i>gui_get_text_bg_color</i> .....	93
8.2.10	<i>gui_clear_dc</i> .....	93
8.2.11	<i>gui_set_pixel</i> .....	94
8.2.12	<i>gui_get_pixel</i> .....	95
8.2.13	<i>gui_text_out</i> .....	95
8.2.14	<i>gui_get_text_width</i> .....	96
8.2.15	<i>gui_get_text_height</i> .....	97
8.2.16	<i>gui_cline</i> .....	97
8.2.17	<i>gui_get_width</i> .....	98
8.2.18	<i>gui_get_height</i> .....	99
8.2.19	<i>gui_page_op_paint</i> .....	99
8.2.20	<i>gui_ime_set_mode</i> .....	100
8.2.21	<i>gui_ime_start_input</i> .....	101
8.2.22	<i>gui_main_menu_func_add</i> .....	101
8.2.23	<i>gui_main_menu_item_add</i> .....	102
8.2.24	<i>gui_main_menu_show</i> .....	103
8.2.25	<i>gui_post_message</i> .....	103
8.2.26	<i>gui_proc_default_msg</i> .....	104
8.2.27	<i>gui_messagebox_show</i> .....	105
8.2.28	<i>gui_load_bmp</i> .....	106
8.2.29	<i>gui_out_bits</i> .....	106
<b>9</b>	<b>EMV(libapi_emv)</b> .....	<b>107</b>
9.1	interface list .....	107
9.2	API interface.....	108
9.2.1	<i>emv_read_card</i> .....	108
9.2.2	<i>EMV_TermConfigInit</i> .....	108
9.2.3	<i>EMV_GetKernelVersion</i> .....	109

---

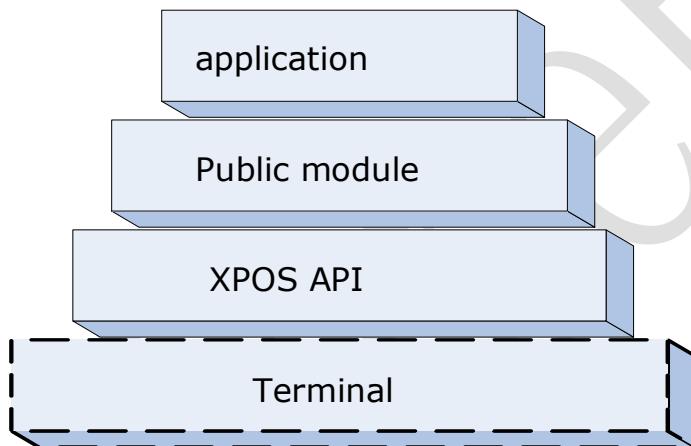
9.2.4	<i>EMV_GetKernelData</i> .....	109
9.2.5	<i>EMV_PrmSetAIDPrm</i> .....	110
9.2.6	<i>EMV_PrmGetAIDPrm</i> .....	111
9.2.7	<i>EMV_PrmDelAIDPrm</i> .....	111
9.2.8	<i>EMV_PrmClearAIDPrmFile</i> .....	112
9.2.9	<i>EMV_PrmSetCAPK</i> .....	112
9.2.10	<i>EMV_PrmGetCAPK</i> .....	113
9.2.11	<i>EMV_PrmDelCAPK</i> .....	113
9.2.12	<i>EMV_PrmClearCAPKFile</i> .....	114
<b>10</b>	<b>Print (libapi_print).....</b>	<b>115</b>
10.1	interface list .....	115
10.2	API interface.....	115
10.2.1	<i>UPrint_GetModuleVer</i> .....	115
10.2.2	<i>UPrint_Init</i> .....	116
10.2.3	<i>UPrint_Str</i> .....	117
10.2.4	<i>UPrint_BitMap</i> .....	117
10.2.5	<i>UPrint_Start</i> .....	118
10.2.6	<i>UPrint_StrBold</i> .....	119
10.2.7	<i>UPrint_Feed</i> .....	119
10.2.8	<i>UPrint_MatrixCode</i> .....	120



# 1 profile

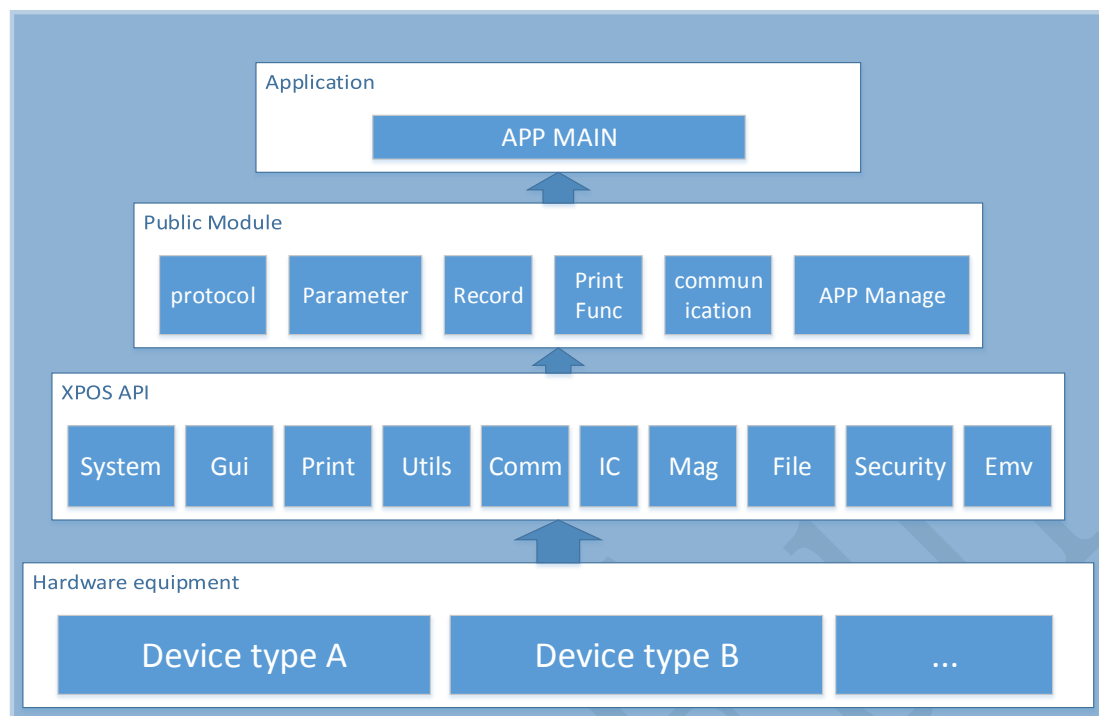
## 1.1 overall structure

This document provides a comprehensive introduction to the application development interface to assist application developers in better secondary development.



## 1.1 modular design

The terminal software is divided into modules to face relatively independent devices or functions, to meet the goal of rapid development of terminal software and frequent update of requirements.



## 2 System module(libapi\_system)

### 2.1 interface list

Function name	Function prototype	Function function
System initialization	Sys_Init	System initialization, independent initialization with application layer
manufacturer personality parameter setting	Sys_Config	Manufacturer personality parameter setting, call each vendor personalization function
Get terminal info	Sys_GetTerminalInfo	Get terminal info
Obtain system time	Sys_GetDateTime	Obtain system time
Set system time	Sys_SetDateTime	Set system time
Scanning buttun	Sys_CheckKey	Scan button, non-blocking

Clear button cache	Sys_ClrKey	Clear button cache
oeprn timer	Sys_TimerOpen	Turn on the timer and set the timing
test timer	Sys_TimerCheck	Check if the timing time is up
Close timer	Sys_TimerClose	Close timer
delay	Sys_Delay	Delay, block
Display battery power in real time	Sys_GetBatter	Display battery power in real time
terminal sleep	Sys_Sleep	Terminal enter into sleep status
terminal reboot	Sys_Reboot	terminal reboot
Get network license for terminal	Sys_GetNetworkLicense	Get network license for terminal
Set screen backlight	Sys_SetScrBackLight	Set screen backlight
Get terminal fireware info	Sys_GetFirmwareInfo	Get terminal fireware info

## 2.2 API interface

### 2.2.1 System initialization(Sys\_Init)

Function prototype	int Sys_Init(int Argc, char **Argv, char *AppName);	
Function function	System initialization, independent initialization with application layer	
Parameter Description	In parameter	Argc: Reuse the main function parameter Argc  Argv: Reuse the main function parameter Argv

		AppName: application name
	Out parameter	none
Return value	USYS_FAIL = -1, // failure  USYS_FIRST = 1, //First run after the program is updated  USYS_NOFIRST = 2 //The program is not the first time running	
Supplementary explanation	no process for not using in parameter  Function internal call vendor private API	
	The API only returns to the first run when the program is newly installed.  The program update is not the first run.	

### 2.2.2 Vendor personality parameter setting(**Sys\_Config**)

Function prototype	void Sys_Config(void);	
Function function	Manufacturer personality parameter setting, call each vendor personalization function	
Parameter description	In parameter	none
	Out parameter	none
Return value		

Supplementary explanation	
------------------------------	--

### 2.2.3 Get terminal info(**Sys\_GetTerminalInfo**)

Function prototype	int Sys_GetTerminalInfo(TERMINALINFO *terminal);	
Function function	Get terminal info	
Parameter description	In parameter	none
	Out parameter	terminal reference TERMINALINFO
Return value	USYS_FAIL = -1, // failure  USYS_SUCCES= 0, // success	
Supplementary explanation		

### 2.2.4 Get system time(**Sys\_GetDateTime**)

Function prototype	int Sys_GetDateTime(byte *DateTime);	
Function function	Get system time	
Parameter	In parameter	None

description	Out parameter	DateTime: "YYYYMMDDHHMMSS" 14 byte
Return value	USYS_FAIL = -1, // failure	
Supplementary explanation	USYS_SUCCES = 0, // success	
Function prototype		

## 2.2.5 Set systemn time(**Sys\_SetDateTime**)

Function prototype	int Sys_SetDateTime(byte *DateTime);	
Function function	Set system time	
Parameter description	In parameter	DateTime: "YYYYMMDDHHMMSS" 14 byte
	Out parameter	None
Return value	USYS_PARAERROR = -2, // parameter wrong  USYS_FAIL = -1, // failure  USYS_SUCCES= 0, // success	
Supplementary explanation	API internal judge whether time format is correct	

## 2.2.6 Scan button(**Sys\_CheckKey**)

Function prototype	int Sys_CheckKey(void);	
Function function	Scan button, non-block	
Parameter description	In parameter	none
	Out parameter	none
Return value	Success return key value KEY_VALUE  No button return 0  USYS_FAIL = -1, // failure	
Supplementary explanation	None enum KEY_VALUE defined ket, unified return 0	

## 2.2.7 Clear button cache(**Sys\_ClrKey**)

Function prototype	void Sys_ClrKey(void);	
Function function	Clear button cache	
Parameter description	In parameter	none

	Out parameter	none
Return value		
Supplementary explanation		

## 2.2.8 Open timer(**Sys\_TimerOpen**)

Function prototype	int Sys_TimerOpen(uint TimerMs);	
Function function	turn on timer, set timer timing	
Parameter description	In parameter	TimerMs: in millisecond
	Out parameter	none
Return value	success return timer handle  USYS_FAIL            = -1,     // failure	
Supplementary explanation		

## 2.2.9 Test timer (**Sys\_TimerCheck**)

Function prototype	int Sys_TimerCheck(int iHandle);
-----------------------	----------------------------------



Function function	test whether timer time is up	
Parameter description	In parameter	iHandle: timer handle
	Out parameter	none
Return value	Successful return, remaining time, in milliseconds (0 means the time is up)	
	USYS_FAIL                      = -1,        // failure	
Supplementary explanation		

## 2.2.10 Turn off timer(**Sys\_TimerClose**)

Function prototype	int Sys_TimerClose(int iHandle);	
Function function	turn off timer	
Parameter description	In parameter	iHandle: timer handle
	Out parameter	none
Return value	USYS_FAIL                      = -1,        // failure	
	USYS_SUCCESS                = 0        // success	
Supplementary explanation		

### 2.2.11 delay(**Sys\_Delay**)

Function prototype	void Sys_Delay(uint uiMs);	
Function function	Postpone, block	
Parameter description	In parameter	uiMs: delay time in ms
	Out parameter	none
Return value		
Supplementary explanation		

### 2.2.12 Terminal sleep (**Sys\_Sleep**)

Function prototype	int Sys_Sleep(uint Time);	
Function function	terminal enter into sleep status	
Parameter description	In parameter	Time: Enter sleep time, in seconds (unsupported manufacturers, considered invalid)
	Out parameter	none
Return value	USYS_FAIL                   = -1,   // failure USYS_SUCCESS               = 0     // success	

Supplementary explanation	The application needs to detect the wireless network registration status after waking up.
---------------------------	---

## 2.2.13 Terminal reboot(**Sys\_Reboot**)

Function prototype	int Sys_Reboot(void);	
Function function	Terminal reboot	
Parameter description	In parameter	none
	Out parameter	none
Return value	USYS_FAIL           = -1,   // failure USYS_SUCCESS       = 0     // success	
Supplementary explanation	Each vendor implements according to its own OS and for unsupport, then direct return. (Considering unsupported vendors, the application layer needs to prompt a forced restart after calling the API)	

# 3 Tool module (**libapi\_util**)

## 3.1 interface list

Function name	Function prototype	Function function
---------------	--------------------	-------------------

ASCII code change to BCD code	Util_Asc2Bcd	ASCII code change to BCD code
BCD code change to ASCII code	Util_Bcd2Asc	BCD code change to ASCII code
Int type data change to BCD code	Util_Int2Bcd	Int type data change to BCD code
BCD code change to int type	Util_Bcd2Int	BCD code data change to int type
Calculate LRC	Util_GenLrc	Calculate and generate LRC check digits (bitwise XOR)
DES encryption and decryption	Util_Des	DES encryption and decryption of data or 3DES encryption and decryption
Chinese character copy	int Util_StrCopy	Chinese intelligent truncation function, solves the problem of displaying half a Chinese character in a line of Chinese
waiting key	Util_WaitKey	Wait for the button within the set time, wait for the timeout without the button
input method	Util_InputMethod	Support data input for input method switching
string input	Util_InputText	Number, letter, password type in
amount input	Util_InputAmount	Input amount
IP input	Util_InputIp	Input IP address
beep	Util_Beep	beep, non-block
Voice play	Play_Voice	voice play, non-block
Production random number	Util_Rand	generate random number


## 3.2 API interface

### 3.2.1 ASCII code change to BCD code

#### (Util\_Asc2Bcd)

Function prototype	int Util_Asc2Bcd(char *AscBuf, char *BcdBuf, int AscLen)	
Function function	ASCII code change to BCD code	
Parameter description	In parameter	AscBuf: ASCII code data to be converted AscLen: Importing ASCII code data length
	Out parameter	BcdBuf: Convert output BCD code data
Return value	UTIL_FAIL = -1, // failure	
	UTIL_SUCCESS = 0 // success	
Supplementary explanation	‘F’ Left on the BCD code, after the number of digits is insufficient, make up ‘F’	

### 3.2.2 BCD code convert to ASCII code

#### (Util\_Bcd2Asc)

Function prototype	int Util_Bcd2Asc(char *BcdBuf, char *AscBuf, int AscLen)	
Function function	BCD code convert to ASCII code	
Parameter description	In parameter	BcdBuf: BCD code data that need to be converted  AscLen: ASCII code data length, which is double the length of BCD code data
	Out parameter	AscBuf: Convert output ASCII code data
Return value	UTIL_FAIL                   = -1,        // failure  UTIL_SUCCESS               = 0         // success	
Supplementary explanation		

### 3.2.3 Int type data convert to BCD code (Util\_Int2Bcd)

Function prototype	int Util_Int2Bcd(uint IntData, char *BcdBuf, int BcdLen)	
Function function	Int type data convert to BCD code	
Parameter description	In parameter	IntData: Int data to be converted  BcdLen: BCD code data length after conversion
	Out parameter	BcdBuf: BCD data after conversion
Return value	UUTIL_FAIL               = -1,               // failure  UUTIL_SUCCESS         = 0               // success	
Supplementary explanation	Right by BCD code, the number of digits is less then add 0 in the left side	

### 3.2.4 BCD code convert to int type(**Util\_Bcd2Int**)

Function prototype	int Util_Bcd2Int(char *BcdBuf, uint *IntData, int BcdLen)	
Function function	BCD code data convert to int type	
Parameter description	In parameter	BcdBuf:BCD data to be converted  BcdLen: BCD code data length
	Out parameter	IntData: int type data after conversion

Return value	UUTIL_FAIL	= -1,	// failure
	UUTIL_SUCCESS	= 0	// success
Supplementary explanation			

### 3.2.5 Caculate LRC(Util\_GenLrc)

Function prototype	Byte Util_GenLrc(char *Data, int DataLen)	
Function function	Calculate and generate LRC check digits (bitwise XOR)	
Parameter description	In parameter	Data: Data of the LRC check digit to be calculated DataLen: data length
	Out parameter	
Return value	Calculate the generated LRC check value	
Supplementary explanation		

### 3.2.6 DES encryption and decryption (Util\_Des)



Function prototype	intUtil_Des(bytebDesType,char*Key,char*InData,char*OutData)	
Function function	DES encryption and decryption of data or 3DES encryption and decryption	
Parameter description	In parameter	bDesType: DES encryption and decryption algorithm: 0 means DES encryption, 1 means DES decryption, 2 means 3DES encryption, 3 means 3DES decryption  Key: The transport key used for encryption and decryption must be a multiple of 8.  InData: The ciphertext data to be encrypted and decrypted must be 8 bytes.
	Out parameter	OutData: The encrypted and decrypted key must be 8 bytes.
Return value	UUTIL_FAIL = -1, // failure	
	UUTIL_SUCCESS = 0 // success	
Supplementary explanation		

### 3.2.7 Chinese character copy(Util\_StrCopy)

Function prototype	int Util_StrCopy(char *dst, cchar *src, int len)	
Function function	Chinese intelligent truncation function, solves the problem of displaying half a Chinese character in a line of Chinese	
Parameter description	In parameter	scr: Source data string  len: Source data length
	Out parameter	Dst: target data string
Return value	Returns the length of the copied string	
Supplementary explanation		

### 3.2.8 **Waiting button(Util\_WaitKey)**

Function prototype	int Util_WaitKey(int TimeOut)	
Function function	Wait for the button within the set time, without button then waiting timeout	
Parameter description	In parameter	TimeOut: Waiting timeout (seconds), 0 means blocking
	Out parameter	

Return value	UUTIL_TIMEOUT Or return button value
Supplementary explanation	

### 3.2.9 Input method input(Util\_InputMethod)

Function prototype	<pre>int Util_InputMethod(int disp_line, char * msgPrompt, int input_line, char *str, int min, int max, byte disp_pattern, int timeout)</pre>	
Function function	data input that support input method switching	
Parameter description	In parameter	<p>disp_line: Prompt message shows the number of lines</p> <p>msgPrompt: Prompt message (left alignment)</p> <p>input_line: Input data display line number</p>
		<p>min: Minimum input length</p> <p>max:Maximum input length</p> <p>disp_pattern: Input data display position, 0 left aligned; 1 centered; 2 right alignment</p> <p>timeout: Waiting for input timeout (seconds)</p>
	Out parameter	str: input data

Return value	<p>Success: return the input data str bytes</p> <p>UTIL_TIMEOUT = -3, // input timeout</p> <p>UTIL_CANCEL = -2, // input cancel</p> <p>UTIL_FAIL = -1, // failure</p>
Supplementary explanation	

### 3.2.10 String input (Util\_InputText)

Function prototype	<pre>int Util_InputText(int disp_line, char *msgPrompt, int input_line, char *str, int min, int max, int disp_pattern, byte disp_mode , int timeout)</pre>	
Function function	number, letter, password input	
Parameter description	In paramater	<p>disp_line: Prompt message shows the number of lines</p> <p>msgPrompt: Prompt message</p> <p>input_line: Input data display line number</p> <p>min: Minimum input length</p> <p>max: Maximum input length</p> <p>disp_pattern: Display position, 0 left aligned; 1 centered; 2, right aligned</p> <p>disp_mode: Input mode, 0 digital input; 1</p>

		number, letter input
		password input
	timeout:	Timeout (seconds)
	Out parameter	str: Input data
Return value	Success: return the input data str bytes  UUTIL_TIMEOUT = -3, // input timeout  UUTIL_CANCEL = -2, // input cancel  UUTIL_FAIL = -1, // failure	
Supplementary explanation	When the input mode is numeric or letter input, switching between a certain number, uppercase and lowercase, and lowercase is performed by pressing a button continuously.	

### 3.2.11 Amount input(Util\_InputAmount)

Function prototype	int Util_InputAmount(int disp_line, char * msgPrompt, int input_line, char *amount, byte disp_ pattern, int timeout)	
Function function	Input amount	
Parameter description	In parameter	disp_line: Prompt message shows the number of lines  msgPrompt: Prompt message  input_line: Input data display line number

### 3.2.12 IP input (Util\_InputIp)

30 / 121

Parameter description	In parameter	disp_line: Prompt message  shows the number of lines  msgPrompt: prompt message  input_line: Input data display line number  disp_pattern: Display position, 0 left aligned; 1 centered; 2, right pair  Timeout: timeout (seconds)
	Out parameter	ip: enter IP address
Return value	Success: return input IP address bytes  UUTIL_TIMEOUT = -3, // input timeout  UUTIL_CANCEL= -2, // input cancel  UUTIL_FAIL = -1, // failure	
Supplementary explanation	API internal with IP address format judgment	

### 3.2.13 beep(Util\_Beep)

Function prototype	Void Util_Beep(int num)
--------------------	-------------------------

Function function	Buzzer, non-blocking	
Parameter description	In paramter	num: Beep times
	Out parameter	
Return value		
Supplementary explanation		

### 3.2.14 **Voice play (Play\_Voice)**

Function prototype	void Play_Voice(char *msg)	
Function function	Specified line display	
Parameter description	In parameter	Msg: Information that requires voice play
	Out parameter	None
Return value	None	
Supplementar y explanation	Voice playback is non-blocking	



### 3.2.15 **Generate random numbers(Util\_Rand)**

Function prototype	int Util_Rand(byte *psRandom)	
Function function	Generate random numbers	
Parameter description	In parameter	
	Out parameter	psRandom8byte Binary random number
Return value	UUTIL_FAIL               = -1,               // failure  UUTIL_SUCCESS       = 0               // success	
Supplementar y explanation	Priority use true random numbers	

## 4 File module(libapi\_file)

### 4.1 Interface list

Function name	Function prototype	function function
Check if the file exists	UFile_Check	Check if the file exists
File open/create	UFile_OpenCreate	File open/create
File reading	UFile_Read	File reading
File writing	UFile_Write	File writing
Positioning file pointer	UFile_Lseek	Positioning file pointer
Delete file record	UFile_Delete	Delete file record
Close file	UFile_Close	Close file
Delete file	UFile_Remove	Delete file
Rename file	UFile_Rename	Rename file
Empty file	UFile_Clear	Empty file content
Take the remaining space of the file system	UFile_GetFreeSpace	Take the remaining space of the file system
Take the number of file records	UFile_GetNumberOfRecords	Take the number of file records
Append file record	UFile_AppendRecord	At the end of the file, add a fixed length record file. Automatically create a file when it does not exist
check records based on index number	UFile_GetRecordByIndex	Find any record by record index
Check record	UFile_GetRecord	Find any record by condition
Update record	UFile_UpdateRecord	Update any record by condition

Update records based on index number	UFile_UpdateRecordByIndex	Update any records based on index number
Delete record	UFile_DeleteRecord	Delete any record by condition
Delete records based on index number	UFile_DeleteRecordByIndex	Delete any record by recording the index number
Read a line of text	UFile_ReadLine	Read a line of text and support \r \n newline
Read non-fixed length records	UFile_ReadTLV	Read non-fixed length record TLV, consistent with IC card TVL format
Write non-fixed data	UFile_WriteTLV	Write non-fixed data TLV
Delete non-fixed record	UFile_DeleteTLV	Delete non-fixed record TLV

## 4.2 API interface

### 4.2.1 Check if the file exists (UFile\_Check)

Function prototype	int UFile_Check(cchar *FileName, int iFileLocation);	
Function function	Check if the file exists	
Parameter description	In parameter	FileName: File name, ending with NULL, up to 16 bytes  iFileLocation: Storage location, see enum

		FILELOCATION
	Out parameter	None
Return value	UFILE_NO_EXIST            = -12, // The specified file does not exist  UFILE_PARAERROR        = -11,            // parameter wrong  UFILE_SUCCESS            = 0                //File operation succeeded	
Supplementary explanation		

#### 4.2.2 File open / create(UFile\_OpenCreate)

Function prototype	intUFile_OpenCreate(cchar*FileName, intiFileLocation, intFlag, FILE_HANDLE *fh, int RecSize);
Function function	File open / create

Parameter description	In parameter	<p>FileName: open file name, end with NULL, maximum 16 bytes in length</p> <p>iFileLocation: storage place, see enum FILELOCATION</p> <p>Flag: Open file mode, the value refer to FileFlags define</p> <p>RecSize: File record size</p> <p>RecSize = 0, Create Open Stream File, Text File</p> <p>RecSize = 1, Create open non-fixed length record (TLV) file</p> <p>8&lt;=RecSize&lt;=4090 Create open fixed length record file</p>
	Out parameter	Fh: File handle
Return value	<p>UFILE_NO_EXIST = -12, //The specified file does not exist</p> <p>UFILE_PARAERROR = -11, //Parameter error</p> <p>UFILE_OPEN_FAIL = -2, //Open error</p> <p>UFILE_FAIL = -1, //File operation failed</p> <p>UFILE_SUCCESS = 0 //File operation succeeded</p>	
Supplementary explanation	<p>RecSize only works for the newly created file</p> <p>For opening a created file, the RecSize parameter should be ignored.</p>	

### 4.2.3 File read(UFile\_Read)

Function prototype	int UFile_Read(FILE_HANDLE handle, char *buffer, int size);	
Function function	File reading	
Parameter description	In parameter	handle: Read file handle  size: Read data size
	Out parameter	buffer: read data
Return value	<p>The file was read successfully: the return value is equal to the number of bytes actually read.</p> <p>UFILE_PARAERROR = -11,                   // paramter error</p> <p>UFILE_READ_FAIL = -5,                   //reading error</p> <p>UFILE_FAIL = -1,                   //file operation failed</p>	
Supplementary explanation		

#### 4.2.4 Write file (UFile\_Write)

Function prototype	int UFile_Write(FILE_HANDLE handle, char *buffer, int size);
--------------------	--

Function function	File writing	
Parameter description	In parameter	<p>handle: Read file handle</p> <p>size: The size of the data to be written</p> <p>buffer: Data to be written</p>
	Out parameter	None
Return value	<p>File write succeeded: the return value is equal to the number of bytes actually written</p> <p>UFILE_PARAERROR = -11,                      // parameter error</p> <p>UFILE_WRITE_FAIL        = -4,                      //write error</p> <p>UFILE_FAIL                = -1,                      //file operation failed</p>	
Supplementar y explanation		

#### 4.2.5 Positioning file pointer(UFile\_Lseek)

Function prototype	<code>long UFile_Lseek(FILE_HANDLE handle, long offset, int origin);</code>
Function function	Positioning file pointer

Parameter description	In parameter	Handle: file handle, offset: offset, origin: starting position, see FileSeekFlags type
	Out parameter	None
Return value	UFILE_PARAERROR = -11,                   //parameter error  UFILE_SEEK_FAIL = -6,                   //Positioning file pointer error  UFILE_SUCCESS = 0	
Supplementary explanation		

#### 4.2.6 Delete file record (UFile\_Delete)

Function prototype	int UFile_Delete(FILE_HANDLE handle, uint size);	
Function function	delete file record	
Parameter description	In parameter	handle: file handle  size: Number of deleted files
	Out parameter	None



Return value	UFILE_PARAERROR       = -11,               //parameter error  UFILE_DELETE_FAIL    = -7,               //Delete file record error  UFILE_SUCCESS         = 0
Supplementary explanation	The specific location of the deletion is determined by the File_Lseek() function.

### 4.2.7 Close file (UFile\_Close)

Function prototype	int UFile_Close(FILE_HANDLE handle);	
Function function	Close file	
Parameter description	In parameter	handle: file handle
	Out parameter	None
Return value	UFILE_PARAERROR   = -11,               // parameter error  UFILE_CLOSE_FAIL    = -8,               //Close file error  UFILE_FAIL           = -1,               //File operation failed  UFILE_SUCCESS       = 0                 //File         operation succeeded	
Supplementary explanation		

## 4.2.8 Delete file (UFile\_Remove)

Function prototype	int UFile_Remove(cchar *filename, int iFileLocation);	
Function function	Delete file	
Parameter description	In paramater	fileName: File name, ending with NULL, up to 16 bytes  iFileLocation: storage location, see enum FILELOCATION
	Out parameter	None
Return value	UFILE_NO_EXIST = -12, //The specified file does not exist  UFILE_PARAERROR = -11, //parameter error  UFILE_FAIL = -1, //File operation failed  UFILE_SUCCESS = 0 //File operation succeed	
Supplementar y explanation		

## 4.2.9 Rename file (UFile\_Rename)

Function prototype	int UFile_Rename(cchar *oldname, int iFileLocation, cchar *newname);	
Function function	Rename file	
Parameter description	In parameter	oldname: old file name iFileLocation: storage location, see enum FILELOCATION newname: new file name
	Out parameter	None
Return value	UFILE_NO_EXIST = -12, //The specified file does not exist  UFILE_PARAERRO = -11, //parameter error R UFILE_FAIL = -1, //File operation failed UFILE_SUCCESS = 0 //File operation succeed	
Supplementary explanation		

#### 4.2.10 Empty file(UFile\_Clear)

Function prototype	int UFile_Clear(cchar *FileName, int iFileLocation);
Function function	Empty file content

Parameter description	In parameter	<p>FileName: File name, ending with NULL, up to 16 bytes</p> <p>iFileLocation: storage location, see enum FILELOCATION</p>
	Out parameter	None
Return value	<p>UFILE_NO_EXIST = -12, //The specified file does not exist</p> <p>UFILE_PARAERROR = -11, //parameter error</p> <p>UFILE_FAIL = -1, //File operation failed</p> <p>UFILE_SUCCESS = 0 //File operation succeed</p>	
Supplementary explanation		

#### 4.2.11 Get free space(UFile\_GetFreeSpace)

Function prototype	long UFile_GetFreeSpace(cchar *drive);
Function function	Take the remaining space of the file system

Parameter description	In parameter	drive: driver name("I:"or"F:")  This parameter should be ignored if there is no driver name
	Out parameter	None
Return value	Success: return the remaining space unit K	
	Failure: UFILE_FAIL = -1, //File operation failed	
Supplementary explanation	Get the size of the remaining space of the drive, regardless of the manufacturer of the drive, no need to deal with the drive, just return the total remaining space of the file system	

## 4.2.12 Get the number of file records (UFile\_GetNumberOfRecords)

Function prototype	int UFile_GetNumberOfRecords(cchar *FileName, int iFileLocation, int Record_Len);	
Function function	Get the number of file records	
Parameter description	In parameter	FileName: file name    iFileLocation: storage location, see enum FILELOCATION  Record_Len: Single record length

	Out parameter	None
Return value	Success: Returns the number of records  failure: UFILE_PARAERROR = -11, //parameter error  UFILE_FAIL = -1, //file operation failed	
Supplementary explanation		

### 4.2.13 Append file record(UFile\_AppendRecord)

Function prototype	int UFile_AppendRecord(cchar *FileName, int iFileLocation, char *Record, int Record_Len);	
Function function	At the end of the file, add a fixed length record file. When the file does not exist, automatically create the file;	
Parameter description	In parameter	FileName: File name, ending with NULL, up to 16 bytes  iFileLocation: storage location, see enum FILELOCATION  Record: record data
		Record_Len: record the length of data
	Out parameter	None
Return value	UFILE_PARAERROR = -11, // parameter error  UFILE_FAIL = -1, // file operation failed	

	UFILE_SUCCESS = 0 //file operation succeed
Supplementary explanation	Power failure protection

#### 4.2.14 Query records based on index number(UFile\_GetRecordByIndex)

Function prototype	int UFile_GetRecordByIndex(cchar *FileName, int iFileLocation, void *Record, int Record_Len, uint Record_Index);	
Function function	Find any record by record index	
Parameter description	In parameter	FileName : file name iFileLocation: storage location, see enum FILELOCATION Record_Len: record length Record_Index : Record index (starting at 0)
	Out parameter	Record: record data

Return value	UFILE_NO_EXIST = -12, //The specified file does not exist  UFILE_PARAERROR = -11, //parameter error  UFILE_NO_RECORD = -10, //record not found  UFILE_READ_FAIL = -5, //reading error  UFILE_OPEN_FAIL = -2, //opening error  UFILE_FAIL = -1, //File operation failed  UFILE_SUCCESS = 0 //File operation succeed
Supplementary explanation	

#### 4.2.15 Check record(UFile\_GetRecord)

Function prototype	int UFile_GetRecord(cchar *FileName, int iFileLocation, void *Record, int Record_Len, DBSEARCOND *Condtion);	
Function function	Find any record by condition	
Parameter description	In parameter	FileName: file name iFileLocation: storage location, see enum FILELOCATION  Record_Len: record length  Condtion: query condition, see DBSearCond structure



	Out parameter	Record: record data
Return value	UFILE_NO_EXIST           = -12, //The specified file does not exist  UFILE_PARAERROR        = -11,            //parameter error  UFILE_NO_RECORD        = -10,            //record not found  UFILE_READ_FAIL        = -5,             //reading error  UFILE_OPEN_FAIL        = -2,             //opening error  UFILE_FAIL              = -1,            //File operation failed  UFILE_SUCCESS           = 0             //File operation succeed	
Supplementar y explanation		

#### 4.2.16 Update record(UFile\_UpdateRecord)

Function prototype	int UFile_UpdateRecord(cchar *FileName, int iFileLocation, void *Record, int Record_Len, DBSEARCOND *Condtion);	
Function function	Update any record by condition	
Parameter description	In parameter	FileName: file name

		<p>iFileLocation: storage location, see enum FILELOCATION</p> <p>Record: record data</p> <p>Record_Len: record length</p> <p>Condition: query condition, see DBSearCond structure</p>
	Out parameter	Record
Return value	<p>UFILE_NO_EXIST = -12, //The specified file does not exist</p> <p>UFILE_PARAERROR = -11, //parameter error</p> <p>UFILE_NO_RECORD = -10, //record not founded UFILE_READ_FAIL</p> <p>= -5, // read error</p> <p>UFILE_WRITE_FAIL = -4, //write error</p> <p>UFILE_OPEN_FAIL = -2, // opening error</p> <p>UFILE_FAIL = -1, //File operation failed</p> <p>UFILE_SUCCESS = 0 //File operation succeed</p>	
Supplementary explanation	<p>Power failure protection</p> <p>Record is both in parameter and out parameter</p> <p>In the case of a successful search, the Record is populated by the search results.</p>	

## 4.2.17 Update records based on index number(UFile\_UpdateRecordByIndex)

Function prototype	intUFile_UpdateRecordByIndex(cchar*FileName,intiFileLocation, void *Record, int Record_Len, uint Index);	
Function function	Update any record by index number	
Parameter description	In parameter	FileName: file name iFileLocation: storage location, see enum FILELOCATION Record: record data
		Record_Len: record length Index: Record index number
	Out parameter	Record
Return value	UFILE_NO_EXIST = -12, //The specified file does not exist UFILE_PARAERROR = -11, //parameter error UFILE_NO_RECORD = -10, //record not founded UFILE_READ_FAIL = -5, // reading error UFILE_WRITE_FAIL = -4, // writing error UFILE_OPEN_FAIL = -2, // opening error UFILE_FAIL = -1, //File operation failed UFILE_SUCCESS = 0 //File operation succeed	

Supplementary explanation	<p>Power failure protection</p> <p>Record is both in parameter and out parameter</p> <p>In the case of a successful search, the Record is populated by the search results.</p>
---------------------------	--

#### 4.2.18 Delete record (UFile\_DeleteRecord)

Function prototype	<pre>int UFile_DeleteRecord(cchar *FileName, int iFileLocation, int Record_Len, DBSEARCOND *Condtion);</pre>	
Function function	Delete any record by condition	
Parameter description	In parameter	<p>FileName: file name iFileLocation: storage location, see enum FILELOCATION</p> <p>Record_Len: record length</p> <p>Condtion: query condition, see DBSearCond structure</p>
	Out parameter	
Return value	<pre>UFILE_NO_EXIST = -12, //The specified file does not exist UFILE_PARAMETER_ERROR = -11, //parameter error UFILE_RECORD_NOT_FOUND = -10, //record not founded</pre>	

	UFILE_DELET E_ UFILE_OPEN_ FA UFILE_FAIL UFILE_SUCCESS	FAIL = -7, IL = -2, = -1, = 0	//Delete file record error //opening error //File operation failed //File operation succeed
Supplementary explanation	Power failure protection		

#### 4.2.19 Delete records based on index number(UFile\_DeleteRecordByIndex)

Function prototype	intUFile_DeleteRecordByIndex(cchar*FileName, intiFileLocation, int Record_Len, uint Index);	
Function function	Delete any record by recording the index number	
Parameter description	In parameter	FileName: file name iFileLocation: storage location, see enum FILELOCATION Record_Len: record length Index: record index number
	Out parameter	

Return value	UFILE_NO_EXIST	= -12,	//specified file not existed
	UFILE_PARAERROR	= -11,	//parameter error
	UFILE_NO_RECORD	= -10,	//record not founded
	UFILE_DELETE_FAIL	= -7,	//Delete file record error
	UFILE_OPEN_FAIL	= -2,	//opening error
	UFILE_FAIL	= -1,	//File operation failed
	UFILE_SUCCESS	= 0	//File operation succeed
Supplementar y explanation	Power failure protection		

#### 4.2.20 Read one line text(UFile\_ReadLine)

Function prototype	int UFile_ReadLine(FILE_HANDLE pFile, char *pLineBuff,uint LineBuffSize);	
Function function	Read a line of text, and support \r \n newline (data read out should not contain newline)	
Parameter description	In parameter	pFile: file handle  LineBuffSize: Buffer size
	Out parameter	pLineBuff : Read text data

Return value	<p>Success: data length</p> <p>UFILE_PARAERROR       = -11,                // parameter error</p> <p>UFILE_READ_FAIL        = -5,                // reading error</p> <p>UFILE_FAIL              = -1,                //file operation failed</p>
Parameter description	For a text file, read a row of data from the current location and jump to the next row.

#### 4.2.21 Read non-fixed length records (UFile\_ReadTLV)

Function prototype	int UFile_ReadTLV(char *FileName, int iFileLocation, uint FldID, char *Data, uint *DataLen);	
Function function	Read non-fixed length record TLV, consistent with IC card TVL format	
Parameter description	In parameter	FileName: file name iFileLocation: storage location, see enum FILELOCATION
		FldID: tag (Tag)
	Out parameter	Data: data (Value)  DataLen: length (length)
Return value	<p>UFILE_NO_EXIST        = -12,                //The specified file does not exist</p> <p>UFILE_PARAERROR       = -11,                // parameter error</p> <p>UFILE_NO_RECORD       = -10,                //record not founded</p>	

	<pre>UFILE_READ_FAIL      = -5,           // reading error  UFILE_OPEN_FAIL      = -2,           //opening error  UFILE_FAIL            = -1,           //File operation failed  UFILE_SUCCESS         = 0            //File operation succeed</pre>
Parameter description	Read the record in TLV format

## 4.2.22 Write non-fixed data(UFile\_WriteTLV)

Function prototype	<pre>int UFile_WriteTLV(char *FileName, int iFileLocation, uint FldID, char *Data, uint *DataLen);</pre>	
Function function	Write non-fixed length record TLV	
Parameter description	In parameter	<p>FileName: file name iFileLocation: storage location, see enum FILELOCATION</p> <p>FldID: tag (Tag)</p> <p>Data: data (Value)</p> <p>DataLen: length (length)</p>
	Out parameter	none
Return value	<pre>UFILE_NO_EXIST      = -12,           //specified file does not existed  UFILE_PARAERROR     = -11,           //parameter error</pre>	
	<pre>UFILE_WRITE_FAIL    = -4,            //writing error</pre>	



	UFILE_OPEN_FAIL       = -2,                        //opening error  UFILE_FAIL             = -1,                     //File operation failed  UFILE_SUCCESS          = 0                        //File operation succeed
Supplementary description	

### 4.2.23 Delete non-fixed length record (UFile\_DeleteTLV)

Function prototype	int UFile_DeleteTLV(char *FileName, int iFileLocation, uint FldID);	
Function function	Delete non-fixed length record TLV	
Parameter description	In parameter	FileName     : file name  iFileLocation: storage location, see enum FILELOCATION  FldID: tag(Tag)
	Out parameter	None

Return value	UFILE_NO_EXIST	= -12,	//specified file does not exist
	UFILE_PARAERROR	= -11,	//parameter error
	UFILE_NO_RECORD	= -10,	//record not founded
	UFILE_DELETE_FAIL	= -7,	//Delete file record error
	UFILE_OPEN_FAIL	= -2,	//opening error
	UFILE_FAIL	= -1,	//File operation failed
	UFILE_SUCCESS	= 0	//File operation succeed
Supplementary description			

## 5 IC card module (libapi\_iccard)

### 5.1 interface list

function name	function prototype	Function function
Turn on IC card device	Icc_Open	Turn on IC card device
Turn off IC card device	Icc_Close	Turn off IC card device
Check the card	Icc_GetCardStatus	Contact card: Check if the card is in the card slot
Contact card powering	Icc_PowerUp	Powering on contact IC card: setting IC card type, card slot category

Contact card power off	Icc_PowerDown	Contact card power off
Contact card communication	Icc_ICComm	Contact IC card communication function
NFC card card search	Icc_CTLSPowerUpAndSee k	NFC card reader search card
NFC card power off	Icc_CTLSPowerDown	NFC card power off
NFC card communication	Icc_CTLSComm	Use APDU to communicate with NFC card

## 5.2 API interface

### 5.2.1 Turn on IC card device (Icc\_Open)

Function prototype	int Icc_Open(int iSlotType);	
Function function	Turn on IC card device	
Parameter description	In paramate	iSlotType: card slot number, see enum SlotType
	Out parameter	None
Return value	UICC_FAIL = -1, // operation failed  UICC_OK = 0 // operation succeed	

Supplementary description	
---------------------------	--

### 5.2.2 Turn off IC card device (Icc\_Close)

Function prototype	int Icc_Close(int iSlotType);	
Function function	Turn off IC card device	
Parameter description	In parameter	iSlotType: card slot number, see enum SlotType
	Out parameter	None
Return value	UICC_FAIL = -1, // operation failed UICC_OK = 0 // operation succeed	
Supplementary description		

### 5.2.3 Turn off IC card device (Icc\_Close)

Function prototype	int Icc_CTLSComm(int iCardType,int iSlotType , ICCAPDU *Apu);
--------------------	---

Function function	Use APDU to communicate with NFC card	
Parameter description	In parameter	<p>iCardType: NFC card type, see enum IccType</p> <p>iSlotType: card slot, see enum SlotType</p> <p>Apdu: refer to ICCAPDU Structure description</p> <p>The various types of card operations are based on the type of OperType operation in the ICCAPDU structure. The data that needs to be passed in during various card operations and the way it is stored in the Apdu structure are discussed separately.</p>
	Out parameter	<p>Apdu: refer to ICCAPDU structure description</p> <p>The returned data is based on the type of OperType operation in the ICCAPDU structure, placed in R_Data</p>
Return value	<p>UICC_COMMAND_FAIL      = -2, // Communication error with card</p> <p>UICC_FAIL                = -1, // operation failed</p> <p>UICC_OK                    = 0 // operation succeed</p>	
Supplement ary description		

### 5.2.4 Test card(Icc\_GetCardStatus)

Function prototype	int Icc_GetCardStatus(int iSlotType);	
Function function	Contact card: Check if the card is in the card slot	
Parameter description	In parameter	iSlotType: card slot number, refer to enum <u>SlotType</u>
	Out parameter	
Return value	UICC_EMPTY     = -3, // no card in card slot	
	UICC_FAIL     = -1, // operation failed	
	UICC_OK        = 0 // operation succeed	
Supplemen tary description	Please call first to open the IC card device (Icc_Open)	

### 5.2.5 Contact card power on(Icc\_PowerUp)

Function prototype	int Icc_PowerUp(int iCardType, int iSlotType);	
Function function	Powering on the contact IC card: Set the IC card type and card slot category.	
Parameter description	In parameter	iCardType: IC card type, see enum <u>IccType</u>  iSlotType: card slot type, refer to enum <u>SlotType</u>
	Out parameter	None
Return value	UICC_EMPTY      = -3, // no card in card slot  UICC_FAIL        = -1, // operation failure  UICC_OK           = 0 // operation succeed	
Supplemen tary description	<u>Contains the card reset operation, and subsequently obtains the card reset information through Icc_GetCardATR</u>	

### 5.2.6 Contact card power off (Icc\_PowerDown)

Function prototype	int Icc_PowerDown(int iCardType , int iSlotType);
Function function	contact card power off

Parameter description	In parameter	iCardType: IC card type, see enum <u>IccType</u>  iSlotType : card slot type , see enum SlotType
	Out parameter	None
Return value	UICC_FAIL      = -1, // operation failure  UICC_OK          = 0 // operation succeed	
Supplementary description	Pay attention to call after power off. Close the IC card device (Icc_Close)	

### 5.2.7 **Contact card communication (Icc\_ICComm)**

Function prototype	int Icc_ICComm (int iCardType,int iSlotType, ICCAPDU *Apu);
Function function	Contact IC card communication function



Parameter description	In parameter	<p>iCardType: IC card type, see enum <u>IccType</u></p> <p>iSlotType : card slot type , see enum <u>SlotType</u></p> <p>Apdu: refer to ICCAPDU structure</p> <p>The various types of card operations are based on the type of OperType operation in the ICCAPDU structure.</p>
		<p>The data that needs to be passed in during various card operations and the way it is stored in the Apdu structure are discussed separately.</p>
	Out parameter	<p>Apdu: refer to ICCAPDU structure</p> <p>The returned data is based on the type of OperType operation in the ICCAPDU structure, placed in R_Data</p>
Return value	<p>UICC_COMMAND_FAIL= -2,// Communication error with card</p> <p>UICC_FAIL = -1,// operation failure</p> <p>UICC_OK = 0// operation succeed</p>	
Supplementary description	None	

## 5.2.8 NFC card searching card (Icc\_CTLSPowerUpAndSeek)

Function prototype	int Icc_CTLSPowerUpAndSeek (int iCardType, int iSlotType, char *psUID);	
Function function	NFC card reader searching card	
Parameter description	In parameter	iCardType : NFC card type, see enum IccType  iSlotType: card slot, see enum SlotType
	Out parameter	psUID: Card serial number, the first byte is the serial number length
Return value	UICC_NORF = -4, // no NFC card  UICC_FAIL = -1, // operation failure  UICC_OK = 0 // operation succeed	
Supplementary description	Please call first to open the IC card device (Icc_Open)  Contains card reset operation application layer loop call  <a href="#">Get card reset information via Icc_GetCardATR</a>	

### 5.2.9 NFC card power off(Icc\_CTLSPowerDown)

Function prototype	int Icc_CTLSPowerDown (int iSlotType);	
Function function	NFC card power off	
Parameter description	In parameter	iSlotType: card slot number, see enum SlotType
	Out parameter	None
Return value	UICC_FAIL = -1, // operation failure  UICC_OK = 0 // operation succeed	
Supplemen tary description	Pay attention to call after power off. Close the IC card device (Icc_Close)	

### 5.2.10 Use APDU to communicate with NFC card(Icc\_CTLSComm)

Function prototype	int Icc_CTLSComm(int iCardType,int iSlotType , ICCAPDU *Apdu);
Function function	use APDU to communicate with NFC card

Parameter description	In parameter	<p>iCardType: NFC card type, see enum IccType</p> <p>iSlotType: card slot, see enum SlotType</p> <p>Apdu : refer to ICCAPDU structure description</p> <p>The various types of card operations are based on the type of OperType operation in the ICCAPDU structure. The data that needs to be passed in during various card operations and the way it is stored in the Apdu structure are discussed separately.</p>
	Out parameter	<p>Apdu : refer to ICCAPDU structure description</p> <p>The returned data is based on the type of OperType operation in the ICCAPDU structure, placed in R_Data</p>
Return value	<p>UICC_COMMAND_FAIL= -2,// communication error with card</p> <p>UICC_FAIL = -1,// operation failure</p> <p>UICC_OK = 0// operation succeed</p>	
Supplementary description		

## 6 communication ( libapi\_comm )

### 6.1 interface list

Function prototype	Function function
comm_net_link	Connect Network
comm_net_unlink	Disconnect from the network
comm_sock_connect	connect to the server
comm_sock_recv	Receive data
comm_sock_send	send data
comm_sock_close	Disconnect the server
comm_ssl_init	ssl initialization
comm_ssl_connect	ssl connect to the server
comm_ssl_send	ssl send data
comm_ssl_recv	ssl Receive data
comm_ssl_close	ssl Disconnect

### 6.2 API interface

#### 6.2.1 comm\_net\_link

Function prototype	<code>int comm_net_link(char * title, char * apn , int timeover);</code>	
Function function	Connect Network	
Parameter description	In parameter	title: Tips for connecting to the network apn: gprs apn timeover : Connection timeout

	Out parameter	
Return value	0,      success Other, failure	
Supplementary description		

### 6.2.2 **comm\_net\_unlink**

Function prototype	int comm_net_unlink();	
Function function	Disconnect from the network	
Parameter description	In parameter	
	Out parameter	
Return value	0,      success Other, failure	
Supplementary description		

### 6.2.3 **comm\_sock\_connect**

Function prototype	int comm_sock_connect(int index, char * ip, int port);	
Function function	Connect to the server	
Parameter description	In parameter	index      sock index ip      server ip port      server port
	Out parameter	
Return value	0,      success Other, failure	
Supplementary description		

### 6.2.4 **comm\_sock\_recv**

Function prototype	int comm_sock_recv(int index, unsigned char * buff, int len, unsigned int timeover);	
Function function	Receive data	
Parameter description	In parameter	index      sock index buff      Receive buffer

		len     Receiving length timeover   overtime time
	Out parameter	
Return value	0,     success Other, failure	
Supplementary description		

### 6.2.5 **comm\_sock\_send**

Function prototype	int comm_sock_send(int index, unsigned char * buff , int size);	
Function function	send data	
Parameter description	In parameter	index     sock index buff     Send buffer len     Send length
	Out parameter	
Return value	0,     success Other, failure	



Supplementary description	
---------------------------	--

## 6.2.6 **comm\_sock\_close**

Function prototype	int comm_sock_close(int index);	
Function function	Disconnect the server	
Parameter description	In parameter	index      sock index
	Out parameter	
Return value	0,      success Other, failure	
Supplementary description		

## 6.2.7 **comm\_ssl\_init**

Function prototype	int comm_ssl_init(int index, char * cacert, char * clientcert, char * clientkey,int level);
Function function	ssl initialization

Parameter description	In parameter	index      sock index cacert      Server certificate clientcert   Client certificate clientkey   Client key level      Verification   level   0=Not verified 1=Verify server certificate
	Out parameter	
Return value	0,      success Other, failure	
Supplementary description		

## 6.2.8 **comm\_ssl\_connect**

Function prototype	int comm_ssl_connect(int index , char * ip , int port);	
Function function	ssl connect to the server	
Parameter description	In parameter	index      sock index ip      server ip port      server port
	Out parameter	

Return value	0, success Other, failure
Supplementary description	

### 6.2.9 **comm\_ssl\_send**

Function prototype	int comm_ssl_send(int index, char * pdata, int size);		
Function function	ssl send data		
Parameter description	In parameter	index data size	sock index ssl data Data size
	Out parameter		
Return value	0, success Other, failure		
Supplementary description			

### 6.2.10 **comm\_ssl\_recv**

Function prototype	int comm_ssl_rcv(int index, char * pdata, int size);		
Function function	ssl Receive data		
Parameter description	In parameter	index data size	sock index ssl data Data size
	Out parameter		
Return value	0, success Other, failure		
Supplementary description			

### 6.2.11 **comm\_ssl\_close**

Function prototype	int comm_ssl_close(int index);		
Function function	ssl Disconnect		
Parameter description	In parameter	index	sock index
	Out parameter		

Return value	0, success Other, failure
Supplementary description	

### 6.2.12 **comm\_wifi\_list\_ap**

Function prototype	int comm_wifi_list_ap(st_wifi_ap_list * ap_list);	
Function function	Get the router list	
Parameter description	In parameter	
	Out parameter	ap_list Router list data, The ap_list space is allocated by the caller with an array size of 10
Return value	Number of routers	
Supplementary description		

### 6.2.13 **comm\_wifi\_link\_ap**

Function prototype	int comm_wifi_link_ap(st_wifi_ap_list * ap_list , char * pwd);	
Function function	Connecting router	
Parameter description	In parameter	ap_list: Router data pwd: password
	Out parameter	
Return value	0, success Other, failure	
Supplementary description		

### 6.2.14 **comm\_wifi\_unlink\_ap**

Function prototype	int comm_wifi_unlink_ap();	
Function function	unlink router	
Parameter description	In parameter	

	Out parameter	ap_listRouter list data, The ap_list space is allocated by the caller with an array size of 10
Return value	0, success Other, failure	
Supplementary description		

### 6.2.15 **comm\_wifi\_get\_link\_state**

Function prototype	int comm_wifi_get_link_state();	
Function function	Get connection status	
Parameter description	In parameter	
	Out parameter	
Return value	1, connection 0, disconnect	
Supplementary description		

## 7 security ( libapi\_security)

### 7.1 interface list

Function prototype	Function function
mksk_save_plaintext_key	Save key plaintext
mksk_save_encrypted_key	Save key ciphertext
mksk_3des_run	Use key 3des operation
dukpt_get_ksn	Get a set of dukpt keys
dukpt_3des_run	Use the previously obtained key 3des operation
dukpt_init_key	Initialize the dukpt key
sec_get_hw_ver	get pci hardware version
sec_get_fw_ver	get pci firmware version

### 7.2 API interface

#### 7.2.1 mksk\_save\_plaintext\_key

Function prototype	int mksk_save_plaintext_key(int type, int gid, unsigned char * key, unsigned char *kvc);
--------------------	--



Function function	Save key plaintext	
Parameter description	In parameter	type:      Key type(0x00-0x04) gid :       Key grouping(0-9) key :       Key plaintext
	Out parameter	kvc   Key      kvc(Key      plaintext encryption 8 0x00)
Return value	0,        success Other, failure	
Supplementary description		

### 7.2.2 msksk\_save\_encrypted\_key

Function prototype	int msksk_save_encrypted_key(int type, int gid, unsigned char * key, unsigned char *kvc);	
Function function	Save key ciphertext	
Parameter description	In parameter	type:      Key type(0x00-0x04) gid :       Key grouping(0-9) key :       Key plaintext
	Out parameter	kvc   Key      kvc(Key      plaintext encryption 8 0x00)

Return value	0, success Other, failure
Supplementary description	

### 7.2.3 mksk\_3des\_run

Function prototype	int mksk_3des_run(int type, int gid, int mode, unsigned char *ind, int size, unsigned char *outd);		
Function function	Use key 3des operation		
Parameter description	In parameter	type: Key type(0x00-0x04) gid : Key grouping(0-9) mode: Operation type (encryption/decryption) ind: Raw data size: Data length (8-byte multiple)	
	Out parameter	outd: Calculation results	
Return value	0, success Other, failure		
Supplementary description			

## 7.2.4 dukpt\_get\_ksn

Function prototype	int dukpt_get_ksn(unsigned char gid, unsigned char * ksn);	
Function function	Get a set of dukpt keys	
Parameter description	In parameter	gid : Key grouping,0
	Out parameter	ksn: Key corresponds to ksn
Return value	0, success Other, failure	
Supplementary description		

## 7.2.5 dukpt\_3des\_run

Function prototype	int dukpt_3des_run(int mode, char *ind, int size, char *outd);	
Function function	Use the previously obtained key 3des operation	
Parameter description	In parameter	mode: Operation type (encryption/decryption) ind: Raw data

		size:      Data length (8-byte multiple)
	Out parameter	outd:      Calculation results
Return value	0,      success Other, failure	
Supplementary description		

### 7.2.6 **dukpt\_init\_key**

Function prototype	int dukpt_init_key(unsigned char gid, unsigned char* init_ksn, unsigned char* init_key);	
Function function	Initialize the dukpt key	
Parameter description	In parameter	gid:      Key grouping init_ksn: Initial ksn init_key: Initial key
	Out parameter	
Return value	0,      success Other, failure	
Supplementary description		

### 7.2.7 **sec\_get\_hw\_ver**

Function prototype	char * sec_get_hw_ver();	
Function function	get pci hardware version	
Parameter description	In parameter	
	Out parameter	
Return value	hardware version	
Supplementary description		

### 7.2.8 **sec\_get\_fw\_ver**

Function prototype	char * sec_get_fw_ver();	
Function function	get pci firmware version	
Parameter description	In parameter	
	Out parameter	

Return value	firmware version
Supplementary description	

## 8 Gui (libapi\_gui)

### 8.1 interface list

Function prototype	Function function
gui_bar_rc	Gui filled area
gui_set_bar_color	Set the fill color
gui_get_bar_color	Get the fill color
gui_set_font	Set display font
gui_get_font	Get display font
gui_set_text_color	Set text color
gui_get_text_color	Get text color
gui_set_text_bg_color	Set the text background color
gui_get_text_bg_color	Get the text background color
gui_clear_dc	Clear screen display
gui_set_pixel	Draw on the screen
gui_get_pixel	The color of the point on the screen
gui_text_out	Display text on the screen

gui_get_text_width	Get the display width of the text
gui_get_text_height	Get the display height of the text
gui_cline	Draw line
gui_get_width	Get screen width
gui_get_height	Get screen height
gui_page_op_paint	Display characters at the bottom left and bottom of the screen
gui_ime_set_mode	Set input method parameters
gui_ime_start_input	Open the input method page
gui_main_menu_func_add	Add menu handler
gui_main_menu_item_add	Add menu item
gui_main_menu_show	Add menu handler
gui_post_message	Send a message
gui_get_message	Recv a message
gui_proc_default_msg	Let the system process the default message
gui_messagebox_show	Display dialog
gui_load_bmp	Load bmp into memory
gui_out_bits	display image

## 8.2 API interface

### 8.2.1 **gui\_bar\_rc**

Function prototype	<code>void gui_bar_rc(int left, int top, int right, int bottom);</code>
Function function	Gui filled area

Parameter description	In parameter	left top right bottom	Left border Upper boundary Right border Lower boundary
	Out parameter		
Return value	0, success Other, failure		
Supplementary description			

### 8.2.2 **gui\_set\_bar\_color**

Function prototype	void gui_set_bar_color(int color);		
Function function	Set the fill color		
Parameter description	In parameter	color	Color format 0x00RRGGBB
	Out parameter		
Return value	0, success Other, failure		



Supplementary description	
---------------------------	--

### 8.2.3 **gui\_get\_bar\_color**

Function prototype	int gui_get_bar_color();	
Function function	Get the fill color	
Parameter description	In parameter	
	Out parameter	
Return value	Fill color	
Supplementary description		

### 8.2.4 **gui\_set\_font**

Function prototype	Set display font
Function function	void gui_set_font(int font);

Parameter description	In parameter	font 0=12 lattice 1=16 lattice
	Out parameter	
Return value		
Supplementary description		

### 8.2.5 **gui\_get\_font**

Function prototype	int gui_get_font(void);	
Function function	Get display font	
Parameter description	In parameter	
	Out parameter	
Return value	Font index	

Supplementary description	
---------------------------	--

### 8.2.6 **gui\_set\_text\_color**

Function prototype	void gui_set_text_color(int color);	
Function function	Set text color	
Parameter description	In parameter	color text color
	Out parameter	
Return value		
Supplementary description		

### 8.2.7 **gui\_get\_text\_color**

Function prototype	int gui_get_text_color(void);
Function function	Get text color

Parameter description	In parameter	
	Out parameter	
Return value	Text color	
Supplementary description		

### 8.2.8 **gui\_set\_text\_bg\_color**

Function prototype	void gui_set_text_bg_color(int color) ;	
Function function	Set the text background color	
Parameter description	In parameter	cloro text color
	Out parameter	
Return value		

Supplementary description	
---------------------------	--

### 8.2.9 **gui\_get\_text\_bg\_color**

Function prototype	int gui_get_text_bg_color(void);	
Function function	Get the text background color	
Parameter description	In parameter	
	Out parameter	
Return value	Text background color	
Supplementary description		

### 8.2.10 **gui\_clear\_dc**

Function prototype	void gui_clear_dc(void);
Function function	Clear screen display

Parameter description	In parameter	
	Out parameter	
Return value		
Supplementary description		

### 8.2.11 **gui\_set\_pixel**

Function prototype	int gui_set_pixel(int x, int y, int color);		
Function function	Draw on the screen		
Parameter description	In parameter	x y color	x coordinate y coordinate Point color
	Out parameter		
Return value	0	success	

Supplementary description	
---------------------------	--

### 8.2.12 **gui\_get\_pixel**

Function prototype	int gui_get_pixel(int x, int y);		
Function function	The color of the point on the screen		
Parameter description	In parameter	x y	x coordinate y coordinate
	Out parameter		
Return value	Point color		
Supplementary description			

### 8.2.13 **gui\_text\_out**

Function prototype	int gui_text_out(int x, int y, char * text);
Function function	Display text on the screen

Parameter description	In parameter	x y text	x coordinate y coordinate Text content
	Out parameter		
Return value	0	success	
Supplementary description			

### 8.2.14 **gui\_get\_text\_width**

Function prototype	int gui_get_text_width(char *text);		
Function function	Get the display width of the text		
Parameter description	In parameter	text	Text content
	Out parameter		
Return value			



Supplementary description	
---------------------------	--

### 8.2.15 **gui\_get\_text\_height**

Function prototype	int gui_get_text_height(char *text);	
Function function	Get the display height of the text	
Parameter description	In parameter	text    Text content
	Out parameter	
Return value	Text height	
Supplementary description		

### 8.2.16 **gui\_cline**

Function prototype	void gui_cline(int x1, int y1, int x2, int y2, int color);
Function function	Draw line

Parameter description	In parameter	x1      Point 1 X coordinate x2      Point 2 X coordinate y1      Point 1 Y coordinate y2      Point 2 Y coordinate color      Line color
	Out parameter	
Return value		
Supplementary description		

### 8.2.17 **gui\_get\_width**

Function prototype	int gui_get_width(void);	
Function function	Get screen width	
Parameter description	In parameter	
	Out parameter	
Return value	Screen width	

Supplementary description	
---------------------------	--

### 8.2.18 **gui\_get\_height**

Function prototype	xxx	
Function function	int gui_get_height(void);	
Parameter description	In parameter	
	Out parameter	
Return value	Screen height	
Supplementary description		

### 8.2.19 **gui\_page\_op\_paint**

Function prototype	void gui_page_op_paint(char * left_str, char * right_str);
Function function	Display characters at the bottom left and bottom of the screen

Parameter description	In parameter	left_str      The character displayed in the lower left corner right_str      The character displayed in the lower right corner
	Out parameter	
Return value		
Supplementary description		

### 8.2.20 **gui\_ime\_set\_mode**

Function prototype	int      gui_ime_set_mode(int      def_mode,      int allow_mode, int password);	
Function function	Set input method parameters	
Parameter description	In parameter	def_mode      Default input method allow_mode      Support input method password enter password
	Out parameter	
Return value		

Supplementary description	
---------------------------	--

### 8.2.21 **gui\_ime\_start\_input**

Function prototype	int gui_ime_start_input(char * buffer, int max, int * position, char * help);		
Function function	Open the input method page		
Parameter description	In parameter	buffer max character position help	Input buffer Maximum  Cursor position Enter page title
	Out parameter		
Return value	Input length		
Supplementary description			

### 8.2.22 **gui\_main\_menu\_func\_add**

Function prototype	int gui_main_menu_func_add(void * pfunc);
--------------------	---

Function function	Add menu handler	
Parameter description	In parameter	pfunc      Menu handler
	Out parameter	
Return value	0 success	
Supplementary description		

### 8.2.23 **gui\_main\_menu\_item\_add**

Function prototype	int gui_main_menu_item_add(st_gui_menu_item_def * menu_item);	
Function function	Add menu item	
Parameter description	In parameter	menu_item      Menu data
	Out parameter	

Return value	0 success
Supplementary description	

### 8.2.24 **gui\_main\_menu\_show**

Function prototype	void gui_main_menu_show(char *id , int timeover);	
Function function	Display menu	
Parameter description	In parameter	id menu id timeover overtime time
	Out parameter	
Return value		
Supplementary description		

### 8.2.25 **gui\_post\_message**

Function prototype	unsigned int gui_post_message(unsigned int msg_id, unsigned int wparam, unsigned int lparam);		
Function function	Send a message		
Parameter description	In parameter	msg_id wparam lparam	Message id parameter 1 parameter 2
	Out parameter		
Return value	0 success		
Supplementary description			

### 8.2.26 **gui\_proc\_default\_msg**

Function prototype	int gui_proc_default_msg( st_gui_message * pmsg );		
Function function	Let the system process the default message		
Parameter description	In parameter	pmsg	Message structure
	Out parameter		



Return value	0 success
Supplementary description	

### 8.2.27 **gui\_messagebox\_show**

Function prototype	int gui_messagebox_show(char *title, char *msg , char* pszLeftOp, char* pszRightOp , int timeover);		
Function function	Display dialog		
Parameter description	In parameter	title msg pszLeftOp pszRightOp corner timeover	Message title Message content Bottom left corner Tip in the lower right overtime time
	Out parameter		
Return value	1 2 3	Confirm return Cancel back Timeout	
Supplementary description			

## 8.2.28 **gui\_load\_bmp**

Function prototype	char * gui_load_bmp(char * filename , int *width , int *height);		
Function function	Load bmp into memory		
Parameter description	In parameter	filename	Image name
	Out parameter	width height	Image width Picture height
Return value	Image content array, which needs to be released after use		
Supplementary description			

## 8.2.29 **gui\_out\_bits**

Function prototype	void gui_out_bits(int x, int y, unsigned char *pbits, int width , int height, int mode);		
Function function	display image		
Parameter description	In parameter	x y pbits	X coordinate Y coordinate Image data

		width height	Image width Picture height
	Out parameter		
Return value			
Supplementary description	Show attention to release pbits		

## 9 EMV(libapi\_emv)

### 9.1 interface list

Function prototype	Function function
emv_read_card	EMV card trans.
EMV_TermConfigInit	Init terminal configure
EMV_GetKernelVersion	EMV kernel version
EMV_GetKernelData	TLV from EMV buffer.
EMV_PrmSetAIDPrm	Save AID buffer.
EMV_PrmGetAIDPrm	Get AID.
EMV_PrmDelAIDPrm	Delete specific AID
EMV_PrmClearAIDPrmFile	Clear all AID.
EMV_PrmSetCAPK	Save CAPK.
EMV_PrmGetCAPK	Get specific CAPK.
EMV_PrmDelCAPK	Delete specific CAPL.
EMV_PrmClearCAPKFile	Clear all CAPK.

## 9.2 API interface

### 9.2.1 **emv\_read\_card**

Function prototype	int emv_read_card(st_read_card_in *card_in, st_read_card_out *card_out);	
Function function	Process of emv card trans.	
Parameter description	In parameter	The parameter of EMV trans.
	Out parameter	Out buffer of EMV trans.
Return value	Result of emv trans.	
Supplementary description		

### 9.2.2 **EMV\_TermConfigInit**

Function prototype	int EMV_TermConfigInit(const TERMCONFIG *ptermconfig);	
Function function	Init terminal configure of emv.	
Parameter description	In parameter	Terminal configure of emv.

	Out parameter	Null
Return value	Result of init terminal configure.	
Supplementary description		

### 9.2.3 **EMV\_GetKernelVersion**

Function prototype	void EMV_GetKernelVersion(char *KernelVersion, int size);	
Function function	Get emv kernel version	
Parameter description	In parameter	Length of version buffer.
	Out parameter	Kernel Version
Return value	Null	
Supplementary description		

### 9.2.4 **EMV\_GetKernelData**

Function prototype	int EMV_GetKernelData (char *Tag, int *Len, byte *Value);
--------------------	---

Function function	Get TLV from EMV buffer.	
Parameter description	In parameter	Tag
	Out parameter	Length    Value
Return value	Result of get TLV data.	
Supplementary description		

### 9.2.5 **EMV\_PrmSetAIDPrm**

Function prototype	int        EMV_PrmSetAIDPrm(TERMINALAPPLIST *pTerminalApps);	
Function function	Set AID buffer into device.	
Parameter description	In parameter	Aid buffer.
	Out parameter	Null
Return value	Result of set aid.	
Supplementary description		

## 9.2.6 **EMV\_PrmGetAIDPrm**

Function prototype	int EMV_PrmGetAIDPrm(TERMINALAPPLIST *pTerminalApps);	
Function function	Get all aid into memory.	
Parameter description	In parameter	Null
	Out parameter	The AID buffer
Return value	Result of get aid buffer.	
Supplementary description		

## 9.2.7 **EMV\_PrmDelAIDPrm**

Function prototype	int EMV_PrmDelAIDPrm(byte *AID, byte AID_Len);	
Function function	Delete the specific AID.	
Parameter description	In parameter	AID Length of AID
	Out parameter	Null

Return value	Result of Delete.
Supplementary description	

### 9.2.8 **EMV\_PrmClearAIDPrmFile**

Function prototype	int EMV_PrmClearAIDPrmFile(void);	
Function function	Clear all AID from device.	
Parameter description	In parameter	Null
	Out parameter	Null
Return value	Result of clear AID.	
Supplementary description		

### 9.2.9 **EMV\_PrmSetCAPK**

Function prototype	int EMV_PrmSetCAPK(CAPUBLICKEY *ppkKey);
Function function	Save CAPK into device.



Parameter description	In parameter	CPAK
	Out parameter	Null
Return value	Result of save CAPK.	
Supplementary description		

### 9.2.10 **EMV\_PrmGetCAPK**

Function prototype	int EMV_PrmGetCAPK(CAPUBLICKEY *ppkKey, byte *RID, byte PKIndex);		
Function function	Get the specific index of CAPK.		
Parameter description	In parameter	RID of CAPK	Index of CAPK
	Out parameter	CAPK	
Return value	Result of get CAPK.		
Supplementary description			

### 9.2.11 **EMV\_PrmDelCAPK**

Function prototype	int EMV_PrmDelCAPK(byte *RID, byte PKIndex);	
Function function	Delete the specific index of CAPK.	
Parameter description	In parameter	RID of CAPK      Index of CAPK
	Out parameter	Null
Return value	Result of delete.	
Supplementary description		

### 9.2.12 **EMV\_PrmClearCAPKFile**

Function prototype	int EMV_PrmClearCAPKFile(void);	
Function function	Clear all CAPK from device.	
Parameter description	In parameter	Null
	Out parameter	Null
Return value	Result of clear.	

Supplementary description	
---------------------------	--

## 10 Print (libapi\_print)

### 10.1 interface list

Function prototype	Function function
UPrint_GetModuleVer	Get version number of print class module
UPrint_Init	Initialize, check the printer status (if it is out of paper), set the print font, use before printing
UPrint_Str	String printing with automatic line break function, support \r, \n newline
UPrint_BitMap	Picture printing
UPrint_Start	Start printing
UPrint_StrBold	String printing (UPrint_StrBold) with automatic line feed function, support \r, \n newline
UPrint_Feed	Printer paper feeding
UPrint_MatrixCode	Print QR code

### 10.2 API interface

#### 10.2.1 UPrint\_GetModuleVer

Function prototype	int UPrint_GetModuleVer(char *pszVer);
--------------------	--

Function function	Get version number of print class module	
Parameter description	In parameter	Nothing
	Out parameter	pszVer    Module version number
Return value	> 0 Successfully returns module version number length USYS_FAIL       = -1	
Supplementary description		

### 10.2.2 **UPrint\_Init**

Function prototype	int UPrint_Init(void);	
Function function	Initialize, check the printer status (if it is out of paper), set the print font, use before printing.	
Parameter description	In parameter	Nothing
	Out parameter	Nothing
Return value	UPRN_FILE_FAIL UPRN_OUTOF_PAPER UPRN_DEV_FAIL UPRN_FAIL UPRN_SUCCESS	Fail to open the file The printer is out of paper Printer device failure Printer unknown fault Success

Supplementary description	
---------------------------	--

### 10.2.3 **UPrint\_Str**

Function prototype	int UPrint_Str(char *str, byte attrib, int linegap);	
Function function	String printing with automatic line break function, support \r, \n newline	
Parameter description	In parameter	str: Need to print string information attrib: Font size: 0 small, 1 medium, 2 large linegap: Line spacing: unit pixels, 0 is the default value (for Pin printing use)
	Out parameter	Nothing
Return value	UPRN_CACHE_ERR UPRN_SUCCESS	Save cache failed Success
Supplementary description		

### 10.2.4 **UPrint\_BitMap**

Function prototype	int UPrint_BitMap(char *BmpFile,byte pattern);
Function function	Picture printing

Parameter description	In parameter	BmpFile: Image file name (XXX.bmp) pattern: Alignment: 0 left alignment, 1 center alignment, 2 right alignment
	Out parameter	Nothing
Return value	UPRN_CACHE_ERR UPRN_SUCCESS	Save cache failed Success
Supplementary description		

### 10.2.5 UPrint\_Start

Function prototype	int UPrint_Start(void);	
Function function	Start printing	
Parameter description	In parameter	Nothing
	Out parameter	Nothing
Return value	UPRN_HANDLE_BACK UPRN_FILE_FAIL UPRN_LOSE_COMMAND UPRN_OUTOF_PAPER UPRN_DEV_FAIL UPRN_FAIL UPRN_SUCCESS	Split machine handle is not put back Fail to open the file Print handle not obtained The printer is out of paper Printer device failure Printer unknown fault Success

Supplementary description	
---------------------------	--

### 10.2.6 **UPrint\_StrBold**

Function prototype	int UPrint_StrBold(char *pszStr, byte cAttrib, byte cPattern,int nLinegap);	
Function function	String printing with automatic line feed function, support \r, \n newline	
Parameter description	In parameter	pszStr: Need to print string information cAttrib: Font size: 0 small, 1 medium, 2 large cPattern: Print position: 0 left, 1 center, 2 right nlinegap: Line spacing, unit pixels, 0 is the default value (for Pin printing use)
	Out parameter	Nothing
Return value	UPRN_CACHE_ERR UPRN_SUCCESS	Save cache failed Success
Supplementary description		

### 10.2.7 **UPrint\_Feed**

Function prototype	int UPrint_Feed(int nFeedLines);
--------------------	----------------------------------

Function function	Printer paper feeding	
Parameter description	In parameter	nFeedLines    Paper length (pixels)
	Out parameter	Nothing
Return value	UPRN_CACHE_ERR            Save cache failed UPRN_SUCCESS            Success	
Supplementary description		

### 10.2.8 **UPrint\_MatrixCode**

Function prototype	int UPrint_MatrixCode(const char *psMatrixCode, int nLen,byte cSize,byte cPattern);	
Function function	Print QR code ( UPrint_MatrixCode ) ,Convert incoming data to QR code and print	
Parameter description	In parameter	psMatrixCode: QR code data nLen:        QR code data length cSize:        QR code size, 0-small, 1-medium, 2-large cPattern:    Alignment, 0 left alignment, 1 center alignment, 2 right alignment
	Out parameter	Nothing



Return value	UPRN_CACHE_ERR UPRN_SUCCESS	Save cache failed Success
Supplementary description		