FKAttend User`s Manual

(Version 2.876)

2011.7

Content

1 Introduction 8

2 FKAttend.OCX Interface 9

2.1 Connection and Disconnection of Devices 9

2.1.1 ConnectComm 9

2.1.2 ConnectNet 9

2.1.3 ConnectUSB 9

2.1.4 DisConnect 10

2.1.5 ConnectGetIP 10

2.2 Management of Registered Data 10

2.2.1 GetEnrollData 10

2.2.2 GetEnrollDataWithString 11

2.2.3 PutEnrollData 11

2.2.4 PutEnrollDataWithString 11

2.2.5 SaveEnrollData 12

2.2.6 DeleteEnrollData 12

2.2.7 USBReadAllEnrollDataFromFile 12

2.2.8 USBReadAllEnrollDataCount 12

2.2.9 USBGetOneEnrollData 13

2.2.10 USBGetOneEnrollDataWithString 13

2.2.11 USBSetOneEnrollData 13

2.2.12 USBSetOneEnrollDataWithString 14

2.2.13 USBWriteAllEnrollDataToFile 14

2.2.14 ReadAllUserID 15

2.2.15 GetAllUserID 15

2.2.16 EmptyEnrollData 15

2.2.17 ClearKeeperData 15

2.2.18 BenumbAllManager 16

2.2.19 GetVerifyMode 16

2.2.20 SetVerifyMode 16

2.2.21 USBGetOneEnrollData_1 16

2.2.22 USBGetOneEnrollDataWithString_1 17

2.2.23 USBSetOneEnrollData_1 17

2.2.24 USBSetOneEnrollDataWithString_1 18

2.2.25 USBReadAllEnrollDataFromFile_Color 18

2.2.26 USBWriteAllEnrollDataToFile_Color 18

2.2.27 USBGetOneEnrollData_Color 19

2.2.28 USBGetOneEnrollDataWithString_Color 19

2.2.29 USBSetOneEnrollData_Color 20
2.2.30 USBSetOneEnrollDataWithString_Color 20
2.3 Management of Recorded Data 21
2.3.1 LoadSuperLogData 21
2.3.2 USBLoadSuperLogDataFromFile 21
2.3.3 GetSuperLogData 21
2.3.4 EmptySuperLogData 22
2.3.5 LoadGeneralLogData 22
2.3.6 USBLoadGeneralLogDataFromFile 22
2.3.7 GetGeneralLogData 23
2.3.8 EmptyGeneralLogData 24
2.3.9 GetGeneralLogData_1 24
2.3.10 GetSuperLogData_1 26
2.3.11 GetRealTimeInfo 26
2.3.12 SetRealTimeInfo 26
2.4 Management of Registrants` Information 27
2.4.1 EnableUser 27
2.4.2 ModifyPrivilege 27
2.4.3 GetUserName 27
2.4.4 SetUserName 27
2.4.5 GetNewsMessage 28
2.4.6 SetNewsMessage 28
2.4.7 GetUserNewsID 28
2.4.8 SetUserNewsID 28
2.5 Management of Devices 29
2.5.1 EnableDevice 29
2.5.2 PowerOnAllDevice 29
2.5.3 PowerOffDevice 29
2.5.4 GetDeviceTime 29
2.5.5 SetDeviceTime 29
2.5.6 GetDeviceStatus 30
2.5.7 GetDeviceInfo 30
2.5.8 SetDeviceInfo 31
2.5.9 GetProductData 31
2.5.10 GetDeviceVersion 32
2.5.11 GetDeviceTime_1 32
2.5.12 SetDeviceTime_1 32
2.6 Management of Bells 32
2.6.1 GetBellTime 32

2.6.2 GetBellTimeWithString 33
2.6.3 SetBellTime 33
2.6.4 SetBellTimeWithString 33
2.7 Control of Doors 33
2.7.1 GetDoorStatus 33
2.7.2 SetDoorStatus 34
2.7.3 GetPassTime 34
2.7.4 GetPassTimeWithString 34
2.7.5 SetPassTime 35
2.7.6 SetPassTimeWithString 35
2.7.7 GetUserPassTime 35
2.7.8 GetUserPassTimeWithString 35
2.7.9 SetUserPassTime 36
2.7.10 SetUserPassTimeWithString 36
2.7.11 GetGroupPassTime 36
2.7.12 GetGroupPassTimeWithString 37
2.7.13 SetGroupPassTime 37
2.7.14 SetGroupPassTimeWithString 37
2.7.15 GetGroupMatch 37
2.7.16 GetGroupMatchWithString 38
2.7.17 SetGroupMatch 38
2.7.18 SetGroupMatchWithString 38
2.8 Adjust Management 39
2.8.1 GetAdjustInfo 39
2.8.2 SetAdjustInfo 39
2.9 Network Information Management 40
2.9.1 GetServerNetInfo 40
2.9.2 SetServerNetInfo 40
2.9.3 SetUSBModel 41
3 FKAttend.DLL Interface 42
3.1 Connection and Disconnection of Devices 42
3.1.1 FK_ConnectComm 42
3.1.2 FK_ConnectNet 42
3.1.3 FK_ConnectUSB 42
3.1.4 FK_DisConnect 42
3.1.5 FK_ConnectGetIP 43
3.2 Management of Enrollment Data 43
3.2.1 FK_GetEnrollData 43
3.2.2 FK_GetEnrollDataWithString 43

3.2.3 FK_PutEnrollData 43
3.2.4 FK_PutEnrollDataWithString 43
3.2.5 FK_SaveEnrollData....43
3.2.6 FK_DeleteEnrollData 44
3.2.7 FK_USBReadAllEnrollDataFromFile....44
3.2.8 FK_USBReadAllEnrollDataCount....44
3.2.9 FK_USBGetOneEnrollData 44
3.2.10 FK_USBGetOneEnrollDataWithString....44
3.2.11 FK_USBSetOneEnrollData 45
3.2.12 FK_USBSetOneEnrollDataWithString 45
3.2.13 FK_USBWriteAllEnrollDataToFile 45
3.2.14 FK_ReadAllUserID....45
3.2.15 FK_GetAllUserID 45
3.2.16 FK_EmptyEnrollData....45
3.2.17 FK_ClearKeeperData 46
3.2.18 FK_BenumbAllManager 46
3.2.19 FK_GetVerifyMode 46
3.2.20 FK_SetVerifyMode 46
3.2.21 FK_USBGetOneEnrollData_1 46
3.2.22 FK_USBGetOneEnrollDataWithString_1....47
3.2.23 FK_USBSetOneEnrollData_1 47
3.2.24 FK_USBSetOneEnrollDataWithString_1 47
3.2.25 FK_USBReadAllEnrollDataFromFile_Color....47
3.2.26 FK_USBWriteAllEnrollDataToFile_Color....47
3.2.27 FK_USBGetOneEnrollData_Color 47
3.2.28 FK_USBGetOneEnrollDataWithString_Color....48
3.2.29 FK_USBSetOneEnrollData_Color....48
3.2.30 FK_USBSetOneEnrollDataWithString_Color 48
3.3 Management of Recorded Data....48
3.3.1 FK_LoadSuperLogData 48
3.3.2 FK_USBLoadSuperLogDataFromFile....48
3.3.3 FK_GetSuperLogData....49
3.3.4 FK_EmptySuperLogData....49
3.3.5 FK_LoadGeneralLogData 49
3.3.6 FK_USBLoadGeneralLogDataFromFile....49
3.3.7 FK_GetGeneralLogData....49
3.3.8 FK_EmptyGeneralLogData....50
3.3.9 FK_GetGeneralLogData_1....50
3.3.10 FK_GetSuperLogData_1....50

3.3.11 FK_GetRealTimeInfo 51
3.3.12 FK_SetRealTimeInfo 51
3.4 Management of Registrant Information 51
3.4.1 FK_EnableUser 51
3.4.2 FK_ModifyPrivilege 51
3.4.3 FK_GetUserName 51
3.4.4 FK_SetUserName 51
3.4.5 FK_GetNewsMessage 52
3.4.6 FK_SetNewsMessage 52
3.4.7 FK_GetUserNewsID 52
3.4.8 FK_SetUserNewsID 52
3.5 Management of Device 52
3.5.1 FK_EnableDevice 52
3.5.2 FK_PowerOnAllDevice 52
3.5.3 FK_PowerOffDevice 53
3.5.4 FK_GetDeviceTime 53
3.5.5 FK_SetDeviceTime 53
3.5.6 FK_GetDeviceStatus 53
3.5.7 FK_GetDeviceInfo 53
3.5.8 FK_SetDeviceInfo 53
3.5.9 FK_GetProductData 54
3.5.10 FK_GetProductDataWithString 54
3.5.11 FK_GetDeviceVersion 54
3.5.12 FK_GetDeviceTime_1 54
3.5.13 FK_SetDeviceTime_1 54
3.6 Management of Bells 55
3.6.1 FK_GetBellTime 55
3.6.2 FK_GetBellTimeWithString 55
3.6.3 FK_SetBellTime 55
3.6.4 FK_SetBellTimeWithString 55
3.7 Control of Doors 55
3.7.1 FK_GetDoorStatus 55
3.7.2 FK_SetDoorStatus 56
3.7.3 FK_GetPassTime 56
3.7.4 FK_GetPassTimeWithString 56
3.7.5 FK_SetPassTime 56
3.7.6 FK_SetPassTimeWithString 56
3.7.7 FK_GetUserPassTime 56
3.7.8 FK_GetUserPassTimeWithString 57

3.7.9 FK_SetUserPassTime 57
3.7.10 FK_SetUserPassTimeWithString 57
3.7.11 FK_GetGroupPassTime 57
3.7.12 FK_GetGroupPassTimeWithString 57
3.7.13 FK_SetGroupPassTime 58
3.7.14 FK_SetGroupPassTimeWithString 58
3.7.15 FK_GetGroupMatch 58
3.7.16 FK_GetGroupMatchWithString 58
3.7.17 FK_SetGroupMatch 58
3.7.18 FK_SetGroupMatchWithString 58
3.8 Adjust Mangement 60
3.8.1 FK_GetAdjustInfo 60
3.8.2 FK_SetAdjustInfo 60
3.9 Network Information Management 61
3.9.1 GetServerNetInfo 61
3.9.2 SetServerNetInfo 61
3.9.3 SetUSBModel 61
4 Appendix 63
4.1 Structures 63
4.1.1 BELLINFO Structure 63
4.1.2 PASSCTRLTIME Structure 63
4.1.3 USERPASSINFO Structure 64
4.1.4 GROUPPASSINFO Structure 64
4.1.5 GROUPMATCHINFO Structure 64
4.1.6 ADJUSTNFO Structure 64
4.1.7 REALTIMEINFO 结构体 65
4.2 Error Code Table 65

1 Introduction

This manual describes an OEM program product FKAttend which provides interfaces for development of applications using FK6xx serial fingerprint time attendance terminals.

FKAttend consists of FKAttend. ocx, FKAttend.dll and FKViaDev.dll for development of programs.

FKAttend.ocx is an interface OCX for connection of the devices with the applications.

FKAttend.dll is an interface DLL for connection of the devices with the applications. It has the same functions as FKAttend.ocx.

FKViaDev.dll is a communication DLL for communicating with the devices.

The interface is composed of seven parts.

① *Connection and disconnection of devices* – To connect and disconnect with the devices

② *Management of registered data* – To manage the registered data, i.e., to read, write and delete the data of the users(registrants) registered in the devices

③ *Management of recorded data* – To read out the data relating to the management and the attendances recorded in the devices

④ *Management of registrants` information* – To get or set the registrants` names, messages and other information

⑤ *Management of devices* – To get or set the time and status of the devices

⑥ *Management of bells* – To get or set the time of the bells

⑦ *Control of doors* – To get or set the information relating to the control of doors

2 FKAttend.OCX Interface

2.1 Connection and Disconnection of Devices

2.1.1 ConnectComm

Type	**long ConnectComm(long nMachineNumber, long nComPort, long nBaudRate, char * pstrTelNumber, long nWaitDialTime, long nLicense)**	
Functionality	**To open the COM port to connect to the device via the RS-232/485 cable.**	
Parameter	**nMachineNumber**	**Number granted to the device to be connected with**
	nComPort	**Sequence number of COM port**
	nBaudRate	**Communication baudrate**
	pstrTelNumber	**Telephone number**
	nWaitDialTime	**Standby time for phone connection (the unit is ms.)**
	nLicense	**License for connection**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**"pstrTelNumber" and "nWaitDialTime" are used when connecting to the device through the modem. Enter 0 when the modem is not used.**
	2	**"nLicense" is a license number granted to the device for the connection. Enter the correct license nuber, or it is unable to connect with the device.**

2.1.2 ConnectNet

Type	**long ConnectNet(long nMachineNumber, char * strIpAddress, long nPort, long nTimeOut, long nProtocolType, long nNetPassword, long nLicense)**	
Functionality	**To open the network port to connect with the device via the network cable.**	
Parameter	**nMachineNumber**	**Number granted to the device to be connected with**
	strIpAddress	**TCP/IP address of the device to be connected with**
	nPort	**Sequence number of network port**
	nTimeOut	**Standby time for the connection (the unit is ms.)**
	nProtocolType	**Kind of protocol**
	nNetPassword	**Network password**
	nLicense	**License for connection**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**To return error codes after waiting as long as "nTimeOut" designates if the relevant device has not been connected to the network,**
	2	**"nProtocolType" designates the kind of protocol used for the network connection. 0 : PROTOCOL_TCPIP - TCP/IP communication 1 : PROTOCOL_UDP - UDP communication**
	3	**"nLicense" has the same meaning as "2.1.1 ConnectComm".**

2.1.3 ConnectUSB

Type	**long ConnectUSB(long nMachineNumber, long nLicense)**
Functionality	**To open the USB port to connect with the device via the USB cable.**

Parameter	**nMachineNumber**	**Number granted to the device to be connected with**
	nLicense	**License for connection**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**"nLicense" has the same meaning as "2.1.1 ConnectComm".**

2.1.4 DisConnect

Type	**void DisConnect(void)**	
Functionality	**To disconnect with the device**	
Parameter		
Return	**None**	
Others	**1**	**To disconnect with the device linked by ConnectComm or ConnectNet and close the corresponding open ports**

2.1.5 ConnectGetIP

Type	**long ConnectGetIP(BSTR *strComName)**	
Functionality	**Generating IP address by name**	
Parameter	**strComName**	**Name of machine to find its IP address**
Others	**1**	**To disconnect with the device linked by ConnectComm or ConnectNet and close the corresponding open ports**

2.2 Management of Registered Data

2.2.1 GetEnrollData

Type	**long GetEnrollData(long anEnrollNumber, long anBackupNumber, long *apnMachinePrivilege, long *apnEnrollData, long *apnPassWord)**	
Functionality	**To get the authorization and enrollment data of the registrants registered in the device**	
Parameter	**anEnrollNumber**	**Registration number**
	anBackupNumber	**Number representing the kind of the enrollment data**
	apnMachinePrivilege	**Variable pointer to the authorization of the registrants**
	apnEnrollData	**Variable pointer to the fingerprint data**
	apnPassWord	**Variable pointer of data relating to password or cards**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**When the execution successes, the corresponding enrollment data are returned to "apnEnrollData" or "apnPassWord" according to "anBackupNumber".**
	2	**For the meanings of the operational authorization returned as "apnMachinePrivilege", please refer to "2.4.2 ModifyPrivilege".**

	3	Every registrant can have three fingerprints, a password or a card number registered in the devices. The kind of these data is reflected in “anBackupNumber”. The following values are returned to “anBackupNumber”: 0 : BACKUP_FP_0 - registered in the first zone for fingerprints … … 9 : BACKUP_FP_9 - registered in the ninth one 10 : BACKUP_PSW - passwords registered 11 : BACKUP_CARD – cards registered

2.2.2 GetEnrollDataWithString

Type	long GetEnrollDataWithString(long anEnrollNumber, long anBackupNumber, long *apnMachinePrivilege, char * apstrEnrollData)	
Functionality	To get the enrollment data in the type of a string. It is equal to GetEnrollData.	
Parameter	anEnrollNumber	Registration number
	anBackupNumber	Number classifying the kind of the enrollment data
	apnMachinePrivilege	Variable pointer of operational authorization of the registrants
	apstrEnrollData	Variable pointer of the enrollment datas
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	The enrollment data is alwayse returned to “apstrEnrollData”
	2	For other parameters, please refer to “2.2.1 GetEnrollData”.

2.2.3 PutEnrollData

Type	long PutEnrollData(long anEnrollNumber, long anBackupNumber,s long anMachinePrivilege, long *apnEnrollData, long anPassword)	
Functionality	To transmit to the device the enrollment data and operational authorization of the persons to be registereds	
Parameter	anEnrollNumber	Registration number
	anBackupNumber	Number classifying the kind of the enrollment data
	anMachinePrivilege	Operational authorization of the registrant
	apnEnrollData	Variable pointer of the fingerprint data
	anPassword	Password or card number data
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	As for “anBackupNumber”, please refer to “2.2.1 GetEnrollData”.
	2	As for “anMachinePrivilege”, please refer to “2.4.2 ModifyPrivilege”
	3	“apnEnrollData” or “apnPassword” data are transferred according to “anBackupNumber”.
	4	The transferred data will be registered in the device when you should execute the command “SaveEnrollData” after execution of PutEnrollData. For the command “SaveEnrollData”, please refer to “2.2.5 SaveEnrollData”.

2.2.4 PutEnrollDataWithString

Type	long PutEnrollDataWithString(long anEnrollNumber, long anBackupNumber, long anMachinePrivilege, BSTR apstrEnrollData)	
Functionality	To contain the enrollment data in the type of a string. It is equal to PutEnrollData.	
Parameter	anEnrollNumber	Registration number

	anBackupNumber	Number classifying the kind of the enrollment data
	anMachinePrivilege	Operational authorization of the registrants
	apstrEnrollData	Variable pointer of the enrollment data
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	The enrollment data are alwayse are contained by “apstrEnrollData”.
	2	As for the other parameters, please refer to “2.2.3 PutEnrollData”.

2.2.5 SaveEnrollData

Type	long SaveEnrollData(void)	
Functionality	To register in the device the enrollment data transferred with a command “PutEnrollData” or “PutEnrollDataWithString”.	
Parameter		
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	Before using this command, you should transmit to the device the data to be registered with a command “PutEnrollData” or “PutEnrollDataWithString”.

2.2.6 DeleteEnrollData

Type	long DeleteEnrollData(long anEnrollNumber, long anBackupNumber)	
Functionality	To delete the designated enrollment data from the device	
Parameter	anEnrollNumber	Registration number
	anBackupNumber	Number classifying the kind of enrollment data
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	The command fails to be executed if the enrollment data do not exist in the device.

2.2.7 USBReadAllEnrollDataFromFile

Type	long USBReadAllEnrollDataFromFile(char *apstrFilePath)	
Functionality	To read the enrollment data into the internal memory of the PC from the file composed in the USB memory, and analyse them	
Parameter	apstrFilePath	File name
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	The command fails to be executed when the structure of the file is not correct.
	2	To learn the method of using the USB memory in the device, please refer to the relevant user`s manual.

2.2.8 USBReadAllEnrollDataCount

Type	long USBReadAllEnrollDataCount(long *apnValue)	
Functionality	To return into the internal memory of the PC the number of the enrollment data read by using a command “USBReadAllEnrollDataFromFile”.	
Parameter	apnValue	Variable pointer of the enrollment data
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	You should first read the data out with a command USBReadAllEnrollDataFromFile” before executing this command.

2.2.9 USBGetOneEnrollData

Type	**long USBGetOneEnrollData(long *apnEnrollNumber, long *apnBackupNumber, long *apnMachinePrivilege, long *apnEnrollData, long *apnPassWord, long *apnEnableFlag, BSTR *apnEnrollName)**	
Functionality	**To get the enrollment data read with a command "USBReadAllEnrollDataFromFile".**	
Parameter	**apnEnrollNumber**	**Variable pointer of registration numbers**
	apnBackupNumber	**Variable pointer of number classifying the kind of enrollment data**
	apnMachinePrivilege	**Variable pointer of the operational authorization of the registrants**
	apnEnrollData	**Variable pointer of the fingerprint data**
	apnPassWord	**Variable pointer of the password or card number data**
	apnEnableFlag	**Variable pointer of the flag enabling the registrant to use the device**
	apnEnrollName	**Variable pointer of the enroll name**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**This command is similar to "GetEnrollData". The difference is that the former uses USB memories without connecting directly to the device. For the description of "GetEnrollData", please refer to "2.2.1 GetEnrollData".**
	2	**To return a code "RUNERR_LOG_END" after getting all the data**
	3	**The command fails to be executed when there is no enrollment data read into the PC with a command "USBReadAllEnrollDataFromFile".**
	4	**For the meaning of "apnEnableFlag", please refer to "2.4.1 EnableUser".**

2.2.10 USBGetOneEnrollDataWithString

Type	**long USBGetOneEnrollDataWithString(long *apnEnrollNumber, long *apnBackupNumber, long *apnMachinePrivilege, BSTR* apstrEnrollData, long *apnEnableFlag, BSTR *apnEnrollName)**	
Functionality	**To get the enrollment data in the type of a string. It is equal to a "USBGetOneEnrollData".**	
Parameter	**apnEnrollNumber**	**Variable pointer of registration numbers**
	apnBackupNumber	**Variable pointer of number classifying the kind of enrollment data**
	apnMachinePrivilege	**Variable pointer of the operational authorization of the registrants**
	apstrEnrollData	**Variable pointer of the enrollment data**
	apnEnableFlag	**Variable pointer of the flag enabling the registrant to use the device**
	apnEnrollName	**Variable pointer of the enroll name**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**This command is similar to "GetEnrollDataWithString". The difference is that the former uses USB memories without connecting directly to the device. As for "GetEnrollDataWithString", please refer to "2.2.2 GetEnrollDataWithString".**
	2	**As for the others, please refer to "2.2.9 USBGetOneEnrollData".**

2.2.11 USBSetOneEnrollData

Type	**long USBSetOneEnrollData(long anEnrollNumber, long anBackupNumber, long anMachinePrivilege, long *apnEnrollData, long anPassWord, long anEnableFlag, LPCTSTR anEnrollName)**	
Functionality	**To take a form in the internal memory of the PC in order to file the operational authorization and enrollment data of the person to be registered. The file can be used in USB memories.**	
Parameter	**anEnrollNumber**	**Registration number**
	anBackupNumber	**Number classifying the kind of the enrollment data**
	anMachinePrivilege	**Operational authorization of the registrant**
	apnEnrollData	**Variable pointer of the fingerprint data**
	anPassWord	**Password or card number data**
	anEnableFlag	**Flag enabling the registrant to use the device**
	anEnrollName	**Variable pointer of the enroll name**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**This command is similar to "PutEnrollData". The difference is that the former uses USB memories without connecting directly to the device. As for "PutEnrollData", please refer to "2.2.3 PutEnrollData".**
	2	**For the meaning of "anEnableFlag", please refer to "2.4.1 EnableUser".**

2.2.12 USBSetOneEnrollDataWithString

Type	**long USBSetOneEnrollDataWithString(long anEnrollNumber, long anBackupNumber, long anMachinePrivilege, BSTR apstrEnrollData, long anEnableFlag, LPCTSTR anEnrollName)**	
Functionality	**To set the enrollment data in the type of string. This is equal to "USBSetOneEnrollData"**	
Parameter	**anEnrollNumber**	**Registration number**
	anBackupNumber	**Number classifying the kind of the enrollment data**
	anMachinePrivilege	**Operational authorization of the registrant**
	apstrEnrollData	**Variable pointer of the enrollment data**
	anEnableFlag	**Flag enabling the registrant to use the device**
	anEnrollName	**Variable pointer of the enroll name**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**This command is similar to "PutEnrollDataWithString". The difference is that the former uses USB memories without connecting directly to the device. As for "PutEnrollDataWithString", please refer to "2.2.4 PutEnrollDataWithString".**
	2	**As for the others, please refer to "2.2.11 USBSetOneEnrollData".**

2.2.13 USBWriteAllEnrollDataToFile

Type	**long USBWriteAllEnrollDataToFile(char *apstrFilePath)**	
Functionality	**To file the enrollment data formed in the internal memory of the PC by "USBSetOneEnrollData" or "USBSetOneEnrollDataWithString"**	
Parameter	**apstrFilePath**	**File name**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**Before the execution of the command, there should be the data formed by the command "USBSetOneEnrollData" or "USBSetOneEnrollDataWith String".**

	2	For the method of using USB memories in the devices, please refer to the corresponding user`s manuals.

2.2.14 ReadAllUserID

Type	long ReadAllUserID(void)	
Functionality	To read into the internal memory of the PC the information relating to all the registrants enrolled in the device	
Parameter		
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	The read information can be got with a command “GetAllUserID”. As for “GetAllUserID”, please refer to “2.2.15 GetAllUserID”.
	2	The command fails to be executed if the enrolled registrant does not exist.

2.2.15 GetAllUserID

Type	long GetAllUserID(long *apnEnrollNumber, long *apnBackupNumber, long *apnMachinePrivilege, long *apnEnableFlag)	
Functionality	To get one by one the registrants` information read with “ReadAllUserID”.	
Parameter	apnEnrollNumber	Variable pointer of the registration number
	apnBackupNumber	Variable pointer of number classifying the kind of enrollment data
	apnMachinePrivilege	Variable pointer of the operational authorization of the registrant
	apnEnableFlag	Variable pointer of the flag enabling the registrant to use the device
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	The command fails to be executed if there is no registrant`s information read by “ReadAllUserID”.
	2	Code “RUNERR_LOG_END” is returned after the data are all got.
	3	For the meaning of the operational authorization returned with “apnMachinePrivilege”, please refer to “2.4.2 ModifyPrivilege”.
	4	For the meaning of “apnEnableFlag”, please refer to “2.4.1 EnableUser”.

2.2.16 EmptyEnrollData

Type	long EmptyEnrollData(void)	
Functionality	To delete all the registered data from the device	
Parameter		
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	Before the execution of this command, it is necessary to backup the registered data.

2.2.17 ClearKeeperData

Type	long ClearKeeperData(void)	
Functionality	To delete all of the registered and recorded data from the device (it means to initialize the device.)	
Parameter		

Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**Before the execution of this command, it is necessary to backup the registered and recorded data.**

2.2.18 BenumbAllManager

Type	**long BenumbAllManager(void)**	
Functionality	**To delete all the information relating to the administrative authorization in the enrollment data and to set the registrants to general users**	
Parameter		
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	

2.2.19 GetVerifyMode

Type	**long GetVerifyMode(long anEnrollNumber, long *apnVerifyMode)**	
Functionality	**To get veify mode information relating to the users to set the registrants to general users**	
Parameter	**anEnrollNumber**	**Variable of registration numbers**
	apnVerifyMode	**Variable pointer of verify mode of the users**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	

2.2.20 SetVerifyMode

Type	**long SetVerifyMode(long anEnrollNumber, long anVerifyMode)**	
Functionality	**To set veify mode information relating to the users to set the registrants to general users**	
Parameter	**anEnrollNumber**	**Variable of registration numbers**
	anVerifyMode	**Variable of verify mode of the users**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	

2.2.21 USBGetOneEnrollData_1

Type	**long USBGetOneEnrollData_1(long *apnEnrollNumber, long *apnBackupNumber, long *apnVerifyMode, long *apnMachinePrivilege, long *apnEnrollData, long *apnPassWord, long *apnEnableFlag, BSTR *apnEnrollName)**	
Functionality	**To get the enrollment data read with a command “USBReadAllEnrollDataFromFile”.**	
Parameter	**apnEnrollNumber**	**Variable pointer of registration numbers**
	apnBackupNumber	**Variable pointer of number classifying the kind of enrollment data**
	apnVerifyMode	**Variable pointer of verify mode of the users**
	apnMachinePrivilege	**Variable pointer of the operational authorization of the registrants**
	apnEnrollData	**Variable pointer of the fingerprint data**
	apnPassWord	**Variable pointer of the password or card number data**
	apnEnableFlag	**Variable pointer of the flag enabling the registrant to use the device**
	apnEnrollName	**Variable pointer of the enroll name**

Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**This command is similar to "GetEnrollData". The difference is that the former uses USB memories without connecting directly to the device. For the description of "GetEnrollData", please refer to "2.2.1 GetEnrollData".**
	2	**To return a code "RUNERR_LOG_END" after getting all the data**
	3	**The command fails to be executed when there is no enrollment data read into the PC with a command "USBReadAllEnrollDataFromFile".**
	4	**For the meaning of "apnEnableFlag", please refer to "2.4.1 EnableUser".**

2.2.22 USBGetOneEnrollDataWithString_1

Type	**long USBGetOneEnrollDataWithString_1(long *apnEnrollNumber, long *apnBackupNumber, long *apnVerifyMode,long *apnMachinePrivilege, BSTR* apstrEnrollData, long *apnEnableFlag, BSTR *apnEnrollName)**	
Functionality	**To get the enrollment data in the type of a string. It is equal to a "USBGetOneEnrollData".**	
Parameter	**apnEnrollNumber**	**Variable pointer of registration numbers**
	apnBackupNumber	**Variable pointer of number classifying the kind of enrollment data**
	apnVerifyMode	**Variable pointer of verify mode of the users**
	apnMachinePrivilege	**Variable pointer of the operational authorization of the registrants**
	apstrEnrollData	**Variable pointer of the enrollment data**
	apnEnableFlag	**Variable pointer of the flag enabling the registrant to use the device**
	apnEnrollName	**Variable pointer of the enroll name**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**This command is similar to "GetEnrollDataWithString". The difference is that the former uses USB memories without connecting directly to the device. As for "GetEnrollDataWithString", please refer to "2.2.2 GetEnrollDataWithString".**
	2	**As for the others, please refer to "2.2.9 USBGetOneEnrollData".**

2.2.23 USBSetOneEnrollData_1

Type	**long USBSetOneEnrollData_1(long anEnrollNumber, long anBackupNumber, long anVerifyMode,long anMachinePrivilege, long *apnEnrollData, long anPassWord, long anEnableFlag, LPCTSTR anEnrollName)**	
Functionality	**To take a form in the internal memory of the PC in order to file the operational authorization and enrollment data of the person to be registered. The file can be used in USB memories.**	
Parameter	**anEnrollNumber**	**Registration number**
	anBackupNumber	**Number classifying the kind of the enrollment data**
	anVerifyMode	**Variable pointer of verify mode of the users**
	anMachinePrivilege	**Operational authorization of the registrant**
	apnEnrollData	**Variable pointer of the fingerprint data**
	anPassWord	**Password or card number data**
	anEnableFlag	**Flag enabling the registrant to use the device**
	anEnrollName	**Variable pointer of the enroll name**

Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**This command is similar to “PutEnrollData”. The difference is that the former uses USB memories without connecting directly to the device. As for “PutEnrollData”, please refer to “2.2.3 PutEnrollData”.**
	2	**For the meaning of “anEnableFlag”, please refer to “2.4.1 EnableUser”.**

2.2.24 USBSetOneEnrollDataWithString_1

Type	**long USBSetOneEnrollDataWithString_1(long anEnrollNumber, long anBackupNumber, long anVerifyMode, long anMachinePrivilege, BSTR apstrEnrollData, long anEnableFlag, LPCTSTR anEnrollName)**	
Functionality	**To set the enrollment data in the type of string. This is equal to “USBSetOneEnrollData”**	
Parameter	**anEnrollNumber**	**Registration number**
	anBackupNumber	**Number classifying the kind of the enrollment data**
	anVerifyMode	**Variable of verify mode of the users**
	anMachinePrivilege	**Operational authorization of the registrant**
	apstrEnrollData	**Variable pointer of the enrollment data**
	anEnableFlag	**Flag enabling the registrant to use the device**
	anEnrollName	**Variable pointer of the enroll name**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**This command is similar to “PutEnrollDataWithString”. The difference is that the former uses USB memories without connecting directly to the device. As for “PutEnrollDataWithString”, please refer to “2.2.4 PutEnrollDataWithString”.**
	2	**As for the others, please refer to “2.2.11 USBSetOneEnrollData”.**

2.2.25 USBReadAllEnrollDataFromFile_Color

Type	**long USBReadAllEnrollDataFromFile_Color(char *apstrFilePath)**	
Functionality	**To read the enrollment data into the internal memory of the PC from the file composed in the USB memory, and analyse them**	
Parameter	**apstrFilePath**	**File name**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**The command fails to be executed when the structure of the file is not correct.**
	2	**To learn the method of using the USB memory in the device, please refer to the relevant user`s manual.**

2.2.26 USBWriteAllEnrollDataToFile_Color

Type	**long USBWriteAllEnrollDataToFile_Color(char *apstrFilePath, long anNewsKind)**	
Functionality	**To file the enrollment data formed in the internal memory of the PC by “USBSetOneEnrollData” or “USBSetOneEnrollDataWithString”**	
Parameter	**apstrFilePath**	**File name**
	anNewsKind	**News Kind : NewKind = 0x02 : 60 chineses characters NewKind = 0x01 : 24 chineses characters**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	

Others	1	Before the execution of the command, there should be the data formed by the command “USBSetOneEnrollData” or “USBSetOneEnrollDataWith String”.
	2	For the method of using USB memories in the devices, please refer to the corresponding user`s manuals.

2.2.27 USBGetOneEnrollData_Color

Type	long USBGetOneEnrollData_Color(long *apnEnrollNumber, long *apnBackupNumber, long *apnMachinePrivilege, long *apnEnrollData, long *apnPassWord, long *apnEnableFlag, BSTR *apnEnrollName, long anNewsKind)	
Functionality	To get the enrollment data read with a command “USBReadAllEnrollDataFromFile”.	
Parameter	apnEnrollNumber	Variable pointer of registration numbers
	apnBackupNumber	Variable pointer of number classifying the kind of enrollment data
	apnMachinePrivilege	Variable pointer of the operational authorization of the registrants
	apnEnrollData	Variable pointer of the fingerprint data
	apnPassWord	Variable pointer of the password or card number data
	apnEnableFlag	Variable pointer of the flag enabling the registrant to use the device
	apnEnrollName	Variable pointer of the enroll name
	anNewsKind	News Kind : NewKind = 0x02 : 60 chineses characters NewKind = 0x01 : 24 chineses characters
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	This command is similar to “GetEnrollData”. The difference is that the former uses USB memories without connecting directly to the device. For the description of “GetEnrollData”, please refer to “2.2.1 GetEnrollData”.
	2	To return a code “RUNERR_LOG_END” after getting all the data
	3	The command fails to be executed when there is no enrollment data read into the PC with a command “USBReadAllEnrollDataFromFile”.
	4	For the meaning of “apnEnableFlag”, please refer to “2.4.1 EnableUser”.

2.2.28 USBGetOneEnrollDataWithString_Color

Type	long USBGetOneEnrollDataWithString_Color(long *apnEnrollNumber, long *apnBackupNumber, long *apnMachinePrivilege, BSTR* apstrEnrollData, long *apnEnableFlag, BSTR *apnEnrollName, long anNewsKind)	
Functionality	To get the enrollment data in the type of a string. It is equal to a “USBGetOneEnrollData”.	
Parameter	apnEnrollNumber	Variable pointer of registration numbers
	apnBackupNumber	Variable pointer of number classifying the kind of enrollment data
	apnMachinePrivilege	Variable pointer of the operational authorization of the registrants
	apstrEnrollData	Variable pointer of the enrollment data
	apnEnableFlag	Variable pointer of the flag enabling the registrant to use the device
	apnEnrollName	Variable pointer of the enroll name
	anNewsKind	News Kind : NewKind = 0x02 : 60 chineses characters NewKind = 0x01 : 24 chineses characters

Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**This command is similar to “GetEnrollDataWithString”. The difference is that the former uses USB memories without connecting directly to the device. As for “GetEnrollDataWithString”, please refer to “2.2.2 GetEnrollDataWithString”.**
	2	**As for the others, please refer to “2.2.9 USBGetOneEnrollData”.**

2.2.29 USBSetOneEnrollData_Color

Type	**long USBSetOneEnrollData_Color(long anEnrollNumber, long anBackupNumber, long anMachinePrivilege, long *apnEnrollData, long anPassWord, long anEnableFlag, LPCTSTR anEnrollName, long anNewsKind)**	
Functionality	**To take a form in the internal memory of the PC in order to file the operational authorization and enrollment data of the person to be registered. The file can be used in USB memories.**	
Parameter	**anEnrollNumber**	**Registration number**
	anBackupNumber	**Number classifying the kind of the enrollment data**
	anMachinePrivilege	**Operational authorization of the registrant**
	apnEnrollData	**Variable pointer of the fingerprint data**
	anPassWord	**Password or card number data**
	anEnableFlag	**Flag enabling the registrant to use the device**
	anEnrollName	**Variable pointer of the enroll name**
	anNewsKind	**News Kind : NewKind = 0x02 : 60 chineses characters NewKind = 0x01 : 24 chineses characters**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**This command is similar to “PutEnrollData”. The difference is that the former uses USB memories without connecting directly to the device. As for “PutEnrollData”, please refer to “2.2.3 PutEnrollData”.**
	2	**For the meaning of “anEnableFlag”, please refer to “2.4.1 EnableUser”.**

2.2.30 USBSetOneEnrollDataWithString_Color

Type	**long USBSetOneEnrollDataWithString_Color(long anEnrollNumber, long anBackupNumber, long anMachinePrivilege, BSTR apstrEnrollData, long anEnableFlag, LPCTSTR anEnrollName, long anNewsKind)**	
Functionality	**To set the enrollment data in the type of string. This is equal to “USBSetOneEnrollData”**	
Parameter	**anEnrollNumber**	**Registration number**
	anBackupNumber	**Number classifying the kind of the enrollment data**
	anMachinePrivilege	**Operational authorization of the registrant**
	apstrEnrollData	**Variable pointer of the enrollment data**
	anEnableFlag	**Flag enabling the registrant to use the device**
	anEnrollName	**Variable pointer of the enroll name**
	anNewsKind	**News Kind : NewKind = 0x02 : 60 chineses characters NewKind = 0x01 : 24 chineses characters**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	

Others	**1**	**This command is similar to “PutEnrollDataWithString”. The difference is that the former uses USB memories without connecting directly to the device. As for “PutEnrollDataWithString”, please refer to “2.2.4 PutEnrollDataWithString”.**
	2	**As for the others, please refer to “2.2.11 USBSetOneEnrollData”.**

2.3 Management of Recorded Data

2.3.1 LoadSuperLogData

Type	**long LoadSuperLogData(long anReadMark)**	
Functionality	**To read the management data from the device into the internal memory of the PC and analyse them**	
Parameter	**anReadMark**	**Read mark flag**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**The read data can be got by “GetSuperLogData” Please refer to “0 GetSuperLogData”.**
	2	**anReadMark = 1 permits reading the newly-added recorded data alone. anReadMark = 0 permits reading all of the recorded data.**

2.3.2 USBLoadSuperLogDataFromFile

Type	**long USBLoadSuperLogDataFromFile(char *apstrFilePath)**	
Functionality	**To read the management data from the the management data file formed in the USB memory into the internal memory of the PC and analyse them**	
Parameter	**apstrFilePath**	**File name**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**Similar to LoadSuperLogData, this command can be used to get the administrative data when the device has not been connected with the PC.**
	2	**The incorrect structures of the files result in a failure of the execution.**
	3	**For the method of using USB memories in the devices, please refer to the corresponding user`s manual.**

2.3.3 GetSuperLogData

Type	**long GetSuperLogData(long *apnSEnrollNumber, long *apnGEnrollNumber, long *apnManipulation, long *apnBackupNumber, DATE *apnDateTime)**	
Functionality	**To get, one by one, the management data read into the memory of the PC with a command “LoadSuperLogData” or “USBLoadSuperLogDataFromFile”.**	
Parameter	**apnSEnrollNumber**	**Variable pointer of the registration number of the manager**
	apnGEnrollNumber	**Variable pointer of the registration number of the managed**
	apnManipulation	**Variable pointer of the identification number of the managed**

	apnBackupNumber	Variable pointer of the number classifying the kind of the enrollment data of the managed person
	apnDateTime	Variable pointer of the time and the date when the management was recorded
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	After all the data are got, a code “RUNERR_LOG_END” is return.
	2	This command fails to be executed if “LoadSuperLogData” or “USBLoadSuperLogDataFromFile” is not first executed.
	3	The following values are returned to “apnManipulation”: 3 : LOG_ENROLL_USER - To register general users 4 : LOG_ENROLL_MANAGER - To register manager(s) 5 : LOG_ENROLL_DELFP - To delete fingerprint data 6 : LOG_ENROLL_DELPASS - To delete passwords 7 : LOG_ENROLL_DELCARD - To delete card data 8 : LOG_LOG_ALLDEL - To delete all the management data 9 : LOG_SETUP_SYS - To modify the information about the devices 10 : LOG_SETUP_TIME - To modify the time of the devices 11 : LOG_SETUP_LOG - To modify the limit values of the management data 12 : LOG_SETUP_COMM - To modify the communication modes 13 : LOG_PASSTIME - To set the duration for which the doors are passed through 14 : LOG_SETUP_DOOR - To set the information about control of the doors

2.3.4 EmptySuperLogData

Type	long EmptySuperLogData(void)	
Functionality	To delete all the management data from the device	
Parameter		
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	Before the execution of this command, it is necessary to backup the management data.

2.3.5 LoadGeneralLogData

Type	long LoadGeneralLogData(long anReadMark)	
Functionality	To read the attendance data from the device into the internal memory of the PC and make an analysis of them	
Parameter	anReadMark	Read mark flag
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	The read data can be got by “GetGeneralLogData”. Please, refer to “2.3.7 GetGeneralLogData”.
	2	anReadMark = 1 allows to read newly-added recorded data alone. anReadMark = 0 allows to read all the recorded data.

2.3.6 USBLoadGeneralLogDataFromFile

Type	**long USBLoadGeneralLogDataFromFile(BSTR apstrFilePath)**	
Functionality	**To read the recorded data into the internal memory of the PC from the attendance data file formed in the USB memory**	
Parameter	**apstrFilePath**	**File name**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**Similar to “LoadGeneralLogData”, this command can be used to get the attendance data when the device is not connected with the PC.**
	2	**The incorrect structure of the file results in a failure of the execution.**
	3	**For the method of using USB memories in the devices, please refer to the corresponding user`s manual.**

2.3.7 GetGeneralLogData

Type	**long GetGeneralLogData(long *apnEnrolslNumber, long *apnVerifyMode, long *apnInOutMode, DATE *apnDateTime)**	
Functionality	**To get, one by one, the attendance data read in the memory of the PC by a command “LoadGeneralLogData” “USBLoadGeneralLogDataFromFile”.**	
Parameter	**apnEnrollNumber**	**Variable pointer of the registration number of the registrant coming in or going out**
	apnVerifyMode	**Variable pointer of the verification mode**
	apnInOutMode	**Variable pointer of the mode of coming in or going out**
	apnDateTime	**Variable pointer of the time and day when the registrant came in or went out**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**A code “RUNERR_LOG_END” is returned after the data are all got.**

	2	The following values are returned to “apnVerifyMode”: 1 : LOG_FPVERIFY - Verified as fingerprints 2 : LOG_PASSVERIFY - Verified as passwords 3 : LOG_CARDVERIFY - Verified as cards 4 : LOG_FPPASS_VERIFY - Verified as passwords added to fingerprints 5 : LOG_FPCARD_VERIFY - Verified as cards added to fingerprints 6 : LOG_PASSFP_VERIFY - Verified as fingerprints added to passwords 7 : LOG_CARDFP_VERIFY - Verified as fingerprints added to cards (The followings are used in the models with a function of controlling doors. Refer to “2.7 Control of Doors”.) 10 : LOG_OPEN_DOOR - The signal of opening the door is transmitted after the verification. 11 : LOG_CLOSE_DOOR - The signal of closing the door is transmitted after the verification 12 : LOG_OPEN_HAND - The signal of opening the door with the key is transferred. 13 : LOG_OPEN_THREAT - The signal of opening the door by verifying threatened fingerprints is transferred. 14 : LOG_PROG_OPEN - The signal of opening the door is transferred from the controlling device. 15 : LOG_PROG_CLOSE - The signal of closing the door is transferred from the controlling device. 16 : LOG_OPEN_IREGAL - The signal of opening the door is illegally transferred. 17 : LOG_CLOSE_IREGAL - The signal of closing the door is illegally transferred. 18 : LOG_OPEN_COVER - The cover of the device opened 19 : LOG_CLOSE_COVER - The cover of the device closed
	3	This command fails to be executed unless “LoadGeneralLogData” or “USBLoadGeneralLogDataFromFile” is first executed.
	4	The following values are returned to “apnInOutMode”: 0 : LOG_IOMODE_IO - Verified with the general mode 1 : LOG_IOMODE_IN1 - Verified with the mode1 of coming in 2 : LOG_IOMODE_IN2 - Verified with the mode2 of coming in 3 : LOG_IOMODE_IN3 - Verified with the mode3 of coming in 4 : LOG_IOMODE_OUT1 - Verified with the mode1 of going out 5 : LOG_IOMODE_OUT2 - Verified with the mode2 of going out 6 : LOG_IOMODE_OUT3 - Verified with the mode3 of going out

2.3.8 EmptyGeneralLogData

Type	**long EmptyGeneralLogData(void)**	
Functionality	**To delete all the data relating to incoming and outgoing from the device**	
Parameter		
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**It is necessary to backup the data relating to incoming and outgoing before the execution of this command.**

2.3.9 GetGeneralLogData_1

Type	**GetGeneralLogData_1(long *apnEnrollNumber, long pnVerifyMode , long *apnInOutMode, long *apnYear, long *apnMonth, long *apnDay,long *apnHour, long *apnMinute, long *apnSec)**	
Functionality	**To get, one by one, the attendance data read in the memory of the PC by a command "LoadGeneralLogData" "USBLoadGeneralLogDataFromFile".**	
Parameter	**apnEnrollNumber**	**Variable pointer of the registration number of the registrant coming in or going out**
	apnVerifyMode	**Variable pointer of the verification mode**
	apnInOutMode	**Variable pointer of the mode of coming in or going out**
	apnYear,apnMonth apnDay, apnHour apnMinute, apnSec	**Variable pointer of the time and day when the registrant came in or went out**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**A code "RUNERR_LOG_END" is returned after the data are all got.**
	2	**The following values are returned to "apnVerifyMode":** **1 : LOG_FPVERIFY - Verified as fingerprints** **2 : LOG_PASSVERIFY - Verified as passwords** **3 : LOG_CARDVERIFY - Verified as cards** **4 : LOG_FPPASS_VERIFY - Verified as passwords added to fingerprints** **5 : LOG_FPCARD_VERIFY - Verified as cards added to fingerprints** **6 : LOG_PASSFP_VERIFY - Verified as fingerprints added to passwords** **7 : LOG_CARDFP_VERIFY - Verified as fingerprints added to cards** **(The followings are used in the models with a function of controlling doors. Refer to "2.7 Control of Doors".)** **10 : LOG_OPEN_DOOR - The signal of opening the door is transmitted after the verification.** **11 : LOG_CLOSE_DOOR - The signal of closing the door is transmitted after the verification** **12 : LOG_OPEN_HAND - The signal of opening the door with the key is transferred.** **13 : LOG_OPEN_THREAT - The signal of opening the door by verifying threatened fingerprints is transferred.** **14 : LOG_PROG_OPEN - The signal of opening the door is transferred from the controlling device.** **15 : LOG_PROG_CLOSE - The signal of closing the door is transferred from the controlling device.** **16 : LOG_OPEN_IREGAL - The signal of opening the door is illegally transferred.** **17 : LOG_CLOSE_IREGAL - The signal of closing the door is illegally transferred.** **18 : LOG_OPEN_COVER - The cover of the device opened** **19 : LOG_CLOSE_COVER - The cover of the device closed**
	3	**This command fails to be executed unless "LoadGeneralLogData" or "USBLoadGeneralLogDataFromFile" is first executed.**
	4	**The following values are returned to "apnInOutMode":** **0 : LOG_IOMODE_IN - Verified with the mode of coming in** **1 : LOG_IOMODE_OUT - Verified with the mode of going out** **2 : LOG_IOMODE_IO - Verified with the general mode**

2.3.10 GetSuperLogData_1

Type	**long GetSuperLogData_1(long *apnSEnrollNumber, long *apnGEnrollNumber, long *apnManipulation, long *apnBackupNumber, long *apnYear, long *apnMonth, long *apnDay, long *apnHour,long *apnMinute, long *apnSec)**	
Functionality	**To get, one by one, the management data read into the memory of the PC with a command "LoadSuperLogData" or "USBLoadSuperLogDataFromFile".**	
Parameter	**apnSEnrollNumber**	**Variable pointer of the registration number of the manager**
	apnGEnrollNumber	**Variable pointer of the registration number of the managed**
	apnManipulation	**Variable pointer of the identification number of the managed**
	apnBackupNumber	**Variable pointer of the number classifying the kind of the enrollment data of the managed person**
	apnYear, apnMonth apnDay, apnHour apnMinute, apnSec	**Variable pointer of the time and the date when the management was recorded**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**After all the data are got, a code "RUNERR_LOG_END" is return.**
	2	**This command fails to be executed if "LoadSuperLogData" or "USBLoadSuperLogDataFromFile" is not first executed.**
	3	**The following values are returned to "apnManipulation":** **3 : LOG_ENROLL_USER - To register general users** **4 : LOG_ENROLL_MANAGER - To register manager(s)** **5 : LOG_ENROLL_DELFP - To delete fingerprint data** **6 : LOG_ENROLL_DELPASS - To delete passwords** **7 : LOG_ENROLL_DELCARD - To delete card data** **8 : LOG_LOG_ALLDEL - To delete all the management data** **9 : LOG_SETUP_SYS - To modify the information about the devices** **10 : LOG_SETUP_TIME - To modify the time of the devices** **11 : LOG_SETUP_LOG - To modify the limit values of the management data** **12 : LOG_SETUP_COMM - To modify the communication modes** **13 : LOG_PASSTIME - To set the duration for which the doors are passed through** **14 : LOG_SETUP_DOOR - To set the information about control of the doors**

2.3.11 GetRealTimeInfo

Type	**Long GetRealTimeInfo(long* apGetRealTime)**	
Functionality	**To export to the PC the waiting time for transfer of blocks and sectors of time for automatic uploading of transactions**	
Parameter	**apGetRealTime**	**Getting Data**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	

2.3.12 SetRealTimeInfo

Type	**Long SetRealTimeInfo(long* apSetRealTime)**
Functionality	**To write into machines the waiting time for transfer of blocks and sectors of time for automatic uploading of transactions**

Parameter	apSetRealTime	Setting data
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	

2.4 Management of Registrants` Information

2.4.1 EnableUser

Type	long EnableUser(long anEnrollNumber, long anBackupNumber,long anEnableFlag)	
Functionality	To enable/forbid the registrant to use the device	
Parameter	anEnrollNumber	Registration number
	anBackupNumber	Number classifying the kind of the enrollment data
	anEnableFlag	Enabling flas
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	anEnableFlag = 0 stands for impossibility of the use; anEnableFlag = 1 possibility.

2.4.2 ModifyPrivilege

Type	long ModifyPrivilege(long anEnrollNumber, long anBackupNumber, long anMachinePrivilege)	
Functionality	To set the operational authorization of the registrant	
Parameter	anEnrollNumber	Registration number
	anBackupNumber	Number classifying the kind of the enrollment data
	anMachinePrivilege	Operational authorization
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	The registrants can be divided into managers and general users according to the operational authorization. This authorization is reflected in “anMachinePrivilege”. The following values are returned to “anMachinePrivilege”: 0 : MP_NONE - General user (can only be verified through the device.) 1 : MP_ALL - Manager (can operate the device.)

2.4.3 GetUserName

Type	long GetUserName(long anEnrollNumber, char *apstrUserName)	
Functionality	To get the name assigned to the registrant	
Parameter	anEnrollNumber	Registration number
	apstrUserName	Variable pointer containing the name
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	The maximum size of the name contained by “apstrUserName” is 10byte (10 English letters or 5 other letters at most).
	2	The command fails to be executed if no name is assigned.

2.4.4 SetUserName

Type	long SetUserName(long anEnrollNumber, char *apstrUserName)
Functionality	To assign a name to the registrant

Parameter	**anEnrollNumber**	**Registration number**
	apstrUserName	**Variable pointer containing the name**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**The maximum size of the name contained by "apstrUserName" is 10byte (10 English letters or 5 other letters at most).**
	2	**The command fails to be executed if no name is assigned.**

2.4.5 GetNewsMessage

Type	**long GetNewsMessage(long anNewsId, char *apstrNews)**	
Functionality	**To get the designated message from the device**	
Parameter	**anNewsId**	**ID number of the message**
	apstrNews	**Variable pointer of the message data**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**"anNewsId" is a number designating messages. The range is from 0 upto 255.**
	2	**The maximum size of the name contained by "apstrUserName" is 48byte (48 English letters or 24 other letters at most).**

2.4.6 SetNewsMessage

Type	**long SetNewsMessage(long anNewsId, char *apstrNews)**	
Functionality	**To set a message in the device**	
Parameter	**anNewsId**	**ID number of the message**
	apstrNews	**Variable pointer of the message data**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**For the details, please refer to "2.4.5 GetNewsMessage".**

2.4.7 GetUserNewsID

Type	**long GetUserNewsID(long anEnrollNumber, long *apnNewsId)**	
Functionality	**To get the ID number of the message assigned to the registrant**	
Parameter	**anEnrollNumber**	**Registration number**
	apnNewsId	**Variable pointer of the ID number**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**"apnNewsId" is a value to be set under "2.4.6 SetNewsMessage".**

2.4.8 SetUserNewsID

Type	**long SetUserNewsID(long anEnrollNumber, long anNewsId)**	
Functionality	**To assign the registrant the ID number of the message**	
Parameter	**anEnrollNumber**	**Registration number**
	anNewsId	**ID number**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**"apnNewsId" is a value to be set under "2.4.6 SetNewsMessage".**

2.5 Management of Devices

2.5.1 EnableDevice

Type	**long EnableDevice(long anEnabledFlag)**	
Functionality	**To allow/forbid the operation on the device**	
Parameter	**anEnabledFlag**	**Enabling flag**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**It can be used when forbidding the operation on the device for the communication between the PC and the device.**
	2	**anEnabledFlag=0 forbids the operation with a message "Working…" prompted; anEnabledFlag=1 allows it with the normal display shown.**

2.5.2 PowerOnAllDevice

Type	**void PowerOnAllDevice(void)**	
Functionality	**To run the connected devices**	
Parameter		
Return	**None**	
Others	**1**	**This command can be only used with the RS-485 communication.**

2.5.3 PowerOffDevice

Type	**long PowerOffDevice(void)**	
Functionality	**To power off the device**	
Parameter		
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**After the execution of this command, the device is disconnected and powered off.**

2.5.4 GetDeviceTime

Type	**long GetDeviceTime(DATE* apnDateTime)**	
Functionality	**To get the time and date of the device**	
Parameter	**apnDateTime**	**Variable pointer of time and dates**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	

2.5.5 SetDeviceTime

Type	**long SetDeviceTime(DATE anDateTime)**	
Functionality	**To set time and a date on the device**	
Parameter	**apnDateTime**	**Time and date data**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	

2.5.6 GetDeviceStatus

Type	**long GetDeviceStatus(long anStatusIndex, long *apnValue)**	
Functionality	**To get the current status values of the device**	
Parameter	**anStatusIndex**	**ID number of the device status**
	apnValue	**Variable pointer of status values**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**This command helps seize the current status of the device through the PC.**
	2	**The following values are returned to "anStatusIndex":** **1 : GET_MANAGERS - The number of managers existing currently** **2 : GET_USERS - The number of general users existing currently** **3 : GET_FPS - The number of fingerprint data existing currently** **4 : GET_PSWS - The number of password data existing currently** **5 : GET_SLOGS - The number of new management data existing currently** **6 : GET_GLOGS - The number of new Income/Outgoing existing-data.** **7 : GET_ASLOGS - The number of the entire management existing –data.** **8 : GET_AGLOGS - The number of the entire Income/Outgoing existing-data.** **9 : GET_CARDS - The number of card data existing currently**

2.5.7 GetDeviceInfo

Type	**long GetDeviceInfo(long anInfoIndex, long *apnValue)**	
Functionality	**To get the information of the device**	
Parameter	**anInfoIndex**	**ID number of the information about the device**
	apnValue	**Variable pointer of information values**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	

Others	**1**	**The following values are returned to "anInfoIndex":** **1 : DI_MANAGERS - The maximum number of registerable managers** **2 : DI_MACHINENUM - ID number of the device** **3 : DI_LANGAUGE - Language displayed on the device** **4 : DI_POWEROFF_TIME - Auto-poweroff duration** **5 : DI_LOCK_CTRL - Door control flag** **6 : DI_GLOG_WARNING - The number of recorded data generating an alarm against overflow of incoming and outgoing data. When recording data over this value, the alarm rings during the record operation.** **7 : DI_SLOG_WARNING - The number of recorded data generating an alarm against overflow of management data. When recording data over this value, the alarm rings during the record operation** **8 : DI_VERIFY_INTERVALS- Interval for recording verification. Within this time, the repeated verification is not recorded.** **9 : DI_RSCOM_BPS – Baudrate of the serial communication** **Each of the baudrates has the following value.** **BPS_9600 = 3** **BPS_19200 = 4** **BPS_38400 = 5** **BPS_57600 = 6** **BPS_115200 = 7** **10: DI_DATE_SEPARATE- Type of displaying time and dates** **11: DI_VERIFY_KIND: setting of matching modes** **the setting values for matching modes are the followings.** **0: F / P / C** **1: F + P** **2: F + C** **3: C**

2.5.8 SetDeviceInfo

Type	**long SetDeviceInfo(long anInfoIndex, long anValue)**	
Functionality	**To set information in the device**	
Parameter	**anInfoIndex**	**ID number of the information about the device**
	apnValue	**Information values**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**The values of "anInfoIndex" are the same as "2.5.7 GetDeviceInfo" gives.**

2.5.9 GetProductData

Type	**long GetProductData(long anProductIndex, char *apstrProductData)**	
Functionality	**To get the information about the sale of products the seller wrote**	
Parameter	**anProductIndex**	**ID number of the information about the sale**
	apstrProductData	**Variable pointer of the information about the sale**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	

Others	**1**	**The following values are returned to "anProductIndex":** **1 : PRODUCT_SERIALNUMBER - Serial number** **2 : PRODUCT_BACKUPNUMBER - Subscription number** **3 : PRODUCT_CODE - Model number** **4 : PRODUCT_NAME - Model name** **5 : PRODUCT_WEB - Homepage of the seller** **6 : PRODUCT_DATE - Sale date** **7 : PRODUCT_SENDTO - Name of the buyer**

2.5.10 GetDeviceVersion

Type	**long GetDeviceVersion(long *apnVersion)**	
Functionality	**To get the version containing the revision history of every model**	
Parameter	**apnVersion**	**Variable pointer of versions**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	

2.5.11 GetDeviceTime_1

Type	**long GetDeviceTime_1(long *apnYear, long *apnMonth, long *apnDay, long *apnHour, long *apnMinute, long apnSec, long *apnDayOfWeek)**	
Functionality	**To get the time and date of the device**	
Parameter	**apnYear,apnMonth apnDay, apnHour apnMinute,apnSec apnDayOfWeek**	**Variable pointer of time and dates**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	

2.5.12 SetDeviceTime_1

Type	**long SetDeviceTime_1(long anYear, long anMonth, long anDay, long anHour, long anMinute, long anSec, long anDayOfWeek)**	
Functionality	**To set time and a date on the device**	
Parameter	**apnYear,apnMonth apnDay, apnHour apnMinute, apnSec anDayOfWeek**	**Time and date data**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	

2.6 Management of Bells

2.6.1 GetBellTime

Type	**long GetBellTime(long *apnBellCount, long *aptBellInfo)**
Functionality	**To get the information about setting a bell**

Parameter	**apnBellCount**	**Variable pointer of times of the bell ringing**
	aptBellInfo	**Variable pointer of the bell information structure**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**The number of bells ringing at the same time is returned to "apnBellCount".**
	2	**The information about the bell such as the designated number and time is returned to "aptBellInfo". For the meaning, please refer to "4.1.1 BELLINFO Structure".**

2.6.2 GetBellTimeWithString

Type	**long GetBellTimeWithString(long *apnBellCount, char *apstrBellInfo)**	
Functionality	**Equal to a command "GetBellTime", it gets the bell-relating information in the form of strings.**	
Parameter	**apnBellCount**	**Variable pointer of times of a bell ringing**
	apstrBellInfo	**Variable pointer of the string**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**For the details, please refer to "2.6.1 GetBellTime".**

2.6.3 SetBellTime

Type	**long SetBellTime(long anBellCount, long *aptBellInfo)**	
Functionality	**To set the bell-relating information in the device**	
Parameter	**anBellCount**	**Times of a bell ringing**
	aptBellInfo	**Variable pointer of the bell information structure**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**The number of bells ringing at the same time is returned to "apnBellCount".**
	2	**The information about the bell such as the designated number and time is returned to "aptBellInfo". For the meaning, please refer to "4.1.1 BELLINFO Structure".**

2.6.4 SetBellTimeWithString

Type	**long SetBellTimeWithString(long anBellCount, char *apstrBellInfo)**	
Functionality	**Equal to a command "SetBellTime", it sets the bell-relating information in the form of strings.**	
Parameter	**anBellCount**	**Times of a bell ringing**
	apstrBellInfo	**Variable pointer of the bell information structure**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**For the details, please refer to "2.6.3 SetBellTime".**

2.7 Control of Doors

Unsupported in some models.

2.7.1 GetDoorStatus

Type	long GetDoorStatus(long *apnStatusVal)	
Functionality	To get the door opening status	
Parameter	apnStatusVal	Variable pointer of the status value
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	The following values are returned to “apnStatusVal”: 0 : DOOR_CONTROLRESET - control state of door by device. 1 : DOOR_OPEND - Door opened 2 : DOOR_CLOSED – Door closed 3 : DOOR_COMMNAD- by the command for control of doors, door opend for some time and closed.

2.7.2 SetDoorStatus

Type	long SetDoorStatus(long anStatusVal)	
Functionality	To control the door opening status	
Parameter	anStatusVal	Status value
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	for the meanings of “anStatusVal”, refer to “2.7.1 GetDoorStatus”.

2.7.3 GetPassTime

Type	long GetPassTime(long anPassTimeID, long *apnPassTime,long anPassTimeSize)	
Functionality	To get the information about the time zone of opening or closing the door	
Parameter	anPassTimeID	ID number of the information about the time zone
	apnPassTime	Variable pointer of the structure of the above information
	anPassTimeSize	Length of the above structure
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	“anPassTimeID” is a number indicating the structure of the information about the time zone. This value ranges from 0 upto 49, since 50 structures at most can be set.
	2	“apnPassTime” reflects the value of the structure “anPassTimeID” designates. This structure has seven time zones per week. Please refer to “0 PASSCTRLTIME Structure”.
	3	As the length of “apnPassTime”, “anPassTimeSize” helps API decide that the structure is long enough.

2.7.4 GetPassTimeWithString

Type	long GetPassTimeWithString(long anPassTimeID, char *apstrPassTime)	
Functionality	Equal to “GetPassTime”, the information about the time zone is returned into a string.	
Parameter	anPassTimeID	ID number of the information about the time zone
	apnPassTime	Variable pointer of the string of the structure of the above information
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	For the details, please refer to “2.7.3 GetPassTime”.

2.7.5 SetPassTime

Type	**long SetPassTime(long anPassTimeID, long *apnPassTime,long anPassTimeSize)**	
Functionality	**To set the information about the time zone for opening and closing the door**	
Parameter	**anPassTimeID**	**ID number of information about the time zone**
	apnPassTime	**Variable pointer of the structure of the above information**
	anPassTimeSize	**Length of the above structure**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**For the details, please refer to “2.7.3 GetPassTime”.**

2.7.6 SetPassTimeWithString

Type	**long SetPassTimeWithString(long anPassTimeID, char *apstrPassTime)**	
Functionality	**Equal to “SetPassTime”, it contains the information about the time zone in the form of strings.**	
Parameter	**anPassTimeID**	**ID number of information about the time zone**
	apnPassTime	**Variable pointer of the string of the structure of the above information**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**For the details, please refer to “2.7.3 GetPassTime”.**

2.7.7 GetUserPassTime

Type	**long GetUserPassTime(long anEnrollNumber, long *apnGroupID, long *apnPassTimeID, long anPassTimeIDSize)**	
Functionality	**To get the time zone-relaing information group assigned to the designated user and the group assigned individually**	
Parameter	**anEnrollNumber**	**Registration number**
	apnGroupID	**Variable pointer of group number**
	apnPassTimeID	**Variable pointer of the structure of the ID number for the information about the time zone**
	anPassTimeIDSize	**Length of the above structure**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**For the meaning of “apnGroupID”, please refer to “2.7.11 GetGroupPassTime”.**
	2	**“apnPassTimeID” is a array-typed batch structure of ID numbers assigned to the registrants. For its definition, please refer to “4.1.3 USERPASSINFO Structure”; for the meanings of the ID numbers, refer to “2.7.3 GetPassTime”.**
	3	**As the length of “apnPassTime”, “anPassTimeSize” helps API determine whether the structure is long enough.**

2.7.8 GetUserPassTimeWithString

Type	**long GetUserPassTimeWithString(long anEnrollNumber, long *apnGroupID, char *apstrPassTimeID)**	
Functionality	**Equal to “GetUserPassTime”, it returns the structure of ID numbers in the form of strings.**	
Parameter	**anEnrollNumber**	**Registration number**

	apnGroupID	Variable pointer of group numbers
	apstrPassTimeID	Variable pointer of the ID number structure string for the information relating to the time zone
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	For the details, please refer to “2.7.7 GetUserPassTime”.

2.7.9 SetUserPassTime

Type	long SetUserPassTime(long anEnrollNumber, long anGroupID, long *apnPassTimeID, long anPassTimeIDSize)	
Functionality	To set the information group of the time zone and the individually-assigned information for the designated registrant	
Parameter	anEnrollNumber	Registration number
	anGroupID	Group number
	apnPassTimeID	Variable pointer of the ID number structure of the time zone information
	anPassTimeIDSize	Length of the above structure
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	For the details, please refer to “2.7.7 GetUserPassTime”.

2.7.10 SetUserPassTimeWithString

Type	long SetUserPassTimeWithString(long anEnrollNumber, long anGroupID, char *apstrPassTimeID)	
Functionality	Equal to command “SetUserPassTime”, it contains the ID number structure in the form of strings.	
Parameter	anEnrollNumber	Registration number
	anGroupID	Group number
	apstrPassTimeID	Variable pointer of the strings for the ID number structure of the time zone information
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	For the details, please refer to “2.7.7 GetUserPassTime”.

2.7.11 GetGroupPassTime

Type	long GetGroupPassTime(long anGroupID, long *apnPassTimeID, long anPassTimeIDSize)	
Functionality	To get ID numbers of the time zone information corresponding to the designated time zone information group	
Parameter	anGroupID	Group number
	apnPassTimeID	Variable pointer of the ID number structure for the time zone information
	anPassTimeIDSize	Length of the above structure
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	On the devices, structures of time zone information can be used in groups. “anGroupID” is a number indicating the group. It is possible to set five groups at most and this value ranges from 1 upto 5.

	2	“apnPassTimeID” is a array-typed batch structure for time zone information ID numbers assigned to each group. In a group, three ID numbers can be set. For the definition of the structure, please refer to “4.1.4 GROUPPASSINFO Structure”; for the meanings of ID numbers, refer to “2.7.3 GetPassTime”.
	3	As the length of “apnPassTimeID”, “anPassTimeIDSize” helps API determine whether the structure is long enough.

2.7.12 GetGroupPassTimeWithString

Type	long GetGroupPassTimeWithString(long anGroupID, char *apstrPassTimeID)	
Functionality	Equal to “GetGroupPassTime”, it returns the ID number structure in the form of strings.	
Parameter	anGroupID	Group number
	apstrPassTimeID	Variable pointer of the strings for the ID number structure of the time zone information
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	For the details, please refer to “2.7.11 GetGroupPassTime”.

2.7.13 SetGroupPassTime

Type	long SetGroupPassTime(long anGroupID, long *apnPassTimeID, long anPassTimeIDSize)	
Functionality	To set ID numbers of the time zone information in the designated group of the information	
Parameter	anGroupID	Group number
	apnPassTimeID	Variable pointer of the ID number structure of the time zone information
	anPassTimeIDSize	Length of the above structure
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	For the details, please refer to “2.7.11 GetGroupPassTime”.

2.7.14 SetGroupPassTimeWithString

Type	long SetGroupPassTimeWithString(long anGroupID, char *apstrPassTimeID)	
Functionality	Equal to command “SetGroupPassTime”, it contains ID number structures in the form of strings.	
Parameter	anGroupID	Group number
	apstrPassTimeID	Variable pointer of the strings for the ID number structure of the time zone information
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.	
Others	1	For the details, please refer to “2.7.11 GetGroupPassTime”.

2.7.15 GetGroupMatch

Type	long GetGroupMatch(long *apnGroupMatch, long anGroupMatchSize)	
Functionality	To get the door control union of groups of the time zone information structures	
Parameter	apnGroupMatch	Variable pointer of the union structure of groups
	anGroupMatchSize	Length of the above structure

Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**To combine the groups of the time zone information structures and use them for control of the doors (opening or closing doors)** **Ten unions at most can be formed. “apnGroupMatch” is an array-typed batch structure for these unions.** **For the definition of the structure, please refer to “4.1.5 GROUPMATCHINFO Structure”.** **Group numbers are described one after another on the item of structures** **Ex: ‘13’ described if groups No.1 and No.3 are combined at the same time,** **‘135’ described if groups No1, No.3 and No.5 are combined at the same time**
	2	**As the length of “apnPassTimeID”, “anPassTimeIDSize”는 helps API determine whether the structure is long enough.**

2.7.16 GetGroupMatchWithString

Type	**long GetGroupMatchWithString(char *apstrGroupMatch)**	
Functionality	**Equal to command “GetGroupMatchTime”, it returns the union structure in the form of strings.**	
Parameter	**apstrGroupMatch**	**Variable pointer of union structure strings of the groups**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**For the details, please refer to “2.7.15 GetGroupMatch”.**

2.7.17 SetGroupMatch

Type	**long SetGroupMatch(long *apnGroupMatch, long anGroupMatchSize)**	
Functionality	**To set the door control union of groups of the time zone information structures**	
Parameter	**apnGroupMatch**	**Variable pointer of the union structure of groups**
	anGroupMatchSize	**Length of the above structure**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**For the details, please refer to “2.7.15 GetGroupMatch”.**

2.7.18 SetGroupMatchWithString

Type	**long SetGroupMatchWithString(char *apstrGroupMatch)**	
Functionality	**Equal to command “SetGroupMatch”, it contains the union structure in the form of strings.**	
Parameter	**apstrGroupMatch**	**Variable pointer of union structure strings of the groups**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**For the details, please refer to “2.7.15 GetGroupMatch”.**

2.8 Adjust Management

2.8.1 GetAdjustInfo

Type	**long GetAdjustInfo(long* dwAdjustedState, long* dwAdjustedMonth, long* dwAdjustedDay, long* dwAdjustedHour, long* dwAdjustedMinute, long* dwRestoredState, long* dwRestoredMonth, long* dwRestoredDay, long* dwRestoredHour, long* dwRestoredMinute)**	
Functionality	**To get a daylight saving time**	
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
others	**1**	**For details, please refer to 《4.1.6 ADJUSTINFO Structure》.**

2.8.2 SetAdjustInfo

Type	**long SetAdjustInfo(long dwAdjustedState, long dwAdjustedMonth, long dwAdjustedDay, long dwAdjustedHour, long dwAdjustedMinute, long dwRestoredState, long dwRestoredMonth, long dwRestoredDay, long dwRestoredHour, long dwRestoredMinute)**	
Functionality	**To set a daylight saving time**	
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
others	**1**	**For details, please refer to 《4.1.6 ADJUSTINFO Structure》.**

2.9 Network Information Management

2.9.1 GetServerNetInfo

Type	**long GetServerNetInfo(BSTR* astrServerIPAddress, long* apServerPort, long* apServerRequest)**	
Functionality	**To get a server information**	
Parameter	**astrServerIPAddress**	**Server IP Address**
	apServerPort	**Server Port**
	apServerRequest	**Server Flag**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
others		

2.9.2 SetServerNetInfo

Type	**long SetServerNetInfo(LPCTSTR astrServerIPAddress, long anServerPort, long anServerRequest)**	
Functionality	**To set server informationa**	
Parameter	**astrServerIPAddress**	**Server IP Address**
	anServerPort	**Server Port**
	anServerRequest	**Server Flag**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
others		

2.9.3 SetUSBModel

Type	void SetUSBModel(long anModel)	
Functionality	To set machine model info for USB Flash information	
others	1	“anModel” is a machine model info. #define FK625_FP1000 2001 #define FK625_FP2000 2002 #define FK625_FP3000 2003 #define FK625_FP5000 2004 #define FK625_FP10000 2005 #define FK625_FP30000 2006 #define FK625_ID30000 2007 #define FK635_FP700 3001 #define FK635_FP3000 3002 #define FK635_FP10000 3003 #define FK635_ID30000 3004 #define FK723_FP1000 4001 #define FK725_FP1000 5001 #define FK725_FP1500 5002 #define FK725_ID5000 5003 #define FK725_ID30000 5004 #define FK735_FP500 6001 #define FK735_FP3000 6002 #define FK735_ID30000 6003 #define FK925_FP3000 7001 #define FK935_FP3000 8001.

3 FKAttend.DLL Interface

The interfaces of FKAttend.DLL are similar to the ones of FKAttend.OCX.

The important difference lies in returning the ID number of the port connected in the first communication for multi-connections and communicating with the devices by means of this ID number under other commands.

Below are described the commands corresponding in OCX and the differences.

3.1 Connection and Disconnection of Devices

3.1.1 FK_ConnectComm

Type	**long FK_ConnectComm(long nMachineNo, long nComPort, long nBaudRate, char *pstrTelNumber, long nWaitDialTime, long nLicense)**	
Functionality	**To open a COM port to connect with the devices through the RS-232/485 cables**	
Return	**The successful execution returns an ID number, the value greater than 0 indicating the connected port.** **The failure returns the corresponding error code. For the details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**For the details, please refer to "2.1.1 ConnectComm".**

3.1.2 FK_ConnectNet

Type	**long FK_ConnectNet(long nMachineNo, char * pstrIpAddress, long nNetPort, long nTimeOut, long nProtocolType, long nNetPassword, long nLicense)**	
Functionality	**To open a COM port to connect with the devices through the network cable**	
Return	**The successful execution returns an ID number, the value greater than 0 indicating the connected port.** **The failure returns the corresponding error code. For the details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**For the details, please refer to "2.1.2 ConnectNet".**

3.1.3 FK_ConnectUSB

Type	**long FK_ConnectUSB(long nMachineNo, long nLicense)**	
Functionality	**To open a USB port to connect with the devices through the USB cable**	
Return	**The successful execution returns an ID number, the value greater than 0 indicating the connected port.** **The failure returns the corresponding error code. For the details of error codes, please refer to "4.2 Error Code Table".**	
Others	**1**	**For the details, please refer to "2.1.3 ConnectUSB".**

3.1.4 FK_DisConnect

Type	**void FK_DisConnect(long nHandleIndex)**	
Functionality	**To disconnect with the devices**	
Others	**1**	**"nHandleIndex" is an ID number returned by "FK_ConnectComm" or "FK_ConnectNet".**

	2	For the details, please refer to “2.1.4 DisConnect”.

3.1.5 FK_ConnectGetIP

Type	long FK_ConnectGetIP(LPSTR *strComName)	
Functionality	Generating IP address by name	
Parameter	strComName	Name of machine to find its IP address
Others	1	To disconnect with the device linked by ConnectComm or ConnectNet and close the corresponding open ports

3.2 Management of Enrollment Data

3.2.1 FK_GetEnrollData

Type	long FK_GetEnrollData(long nHandleIndex, long nEnrollNumber, long nBackupNumber, long * pnMachinePrivilege, void * pnEnrollData, long * pnPassWord)	
Functionality	To get the operational authorization and enrollment data of the registrant registered on the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.1 GetEnrollData”.

3.2.2 FK_GetEnrollDataWithString

Type	long FK_GetEnrollDataWithString(long nHandleIndex, long nEnrollNumber, long nBackupNumber, long * pnMachinePrivilege, LPSTR *apstrEnrollData)	
Functionality	Equal to command “FK_GetEnrollData”, it gets enrollment data in the form of strings.	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.2 GetEnrollDataWithString”.

3.2.3 FK_PutEnrollData

Type	long FK_PutEnrollData(long nHandleIndex, long nEnrollNumber, long nBackupNumber, long nMachinePrivilege, void * pnEnrollData, long nPassWord)	
Functionality	To transmit to the device the operational authorization and enrollment data of the person to be registered”.	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.3 PutEnrollData”.

3.2.4 FK_PutEnrollDataWithString

Type	long FK_PutEnrollDataWithString(long nHandleIndex, long nEnrollNumber, long nBackupNumber, long nMachinePrivilege, char *apstrEnrollData)	
Functionality	Equal to command “FK_PutEnrollData”, it contains the enrollment data in the form of charater strings.	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.4 PutEnrollDataWithString”.

3.2.5 FK_SaveEnrollData

Type	long FK_SaveEnrollData(long nHandleIndex)

Functionality	**To register on the device the enrollment data transmitted by command “FK_PutEnrollData” or “FK_PutEnrollDataWithString”**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.2.5 SaveEnrollData”.**

3.2.6 FK_DeleteEnrollData

Type	**long FK_DeleteEnrollData(long nHandleIndex, long nEnrollNumber, long nBackupNumber)**	
Functionality	**To delete the designated enrollment data from the device**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.2.6 DeleteEnrollData”.**

3.2.7 FK_USBReadAllEnrollDataFromFile

Type	**long FK_USBReadAllEnrollDataFromFile(long nHandleIndex, char * pstrFilePath)**	
Functionality	**To read the enrollment data from the relevant file formed in the USB memory into the internal memory of the PC and make an analysis of them**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.2.7 USBReadAllEnrollDataFromFile”.**

3.2.8 FK_USBReadAllEnrollDataCount

Type	**long FK_USBReadAllEnrollDataCount(long nHandleIndex, long * pnValue)**	
Functionality	**To return the number of enrollment data read into the memory of the PC with command “FK_USBReadAllEnrollDataFromFile”.**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.2.8 USBReadAllEnrollDataCount”.**

3.2.9 FK_USBGetOneEnrollData

Type	**long FK_USBGetOneEnrollData(long nHandleIndex, long * pnEnrollNumber, long * pnBackupNumber, long * pnMachinePrivilege, void * pnEnrollData, long * pnPassWord, long * pnEnableFlag, LPSTR * dwEnrollName)**	
Functionality	**To get the enrollment data read with command “USBReadAllEnrollDataFromFile”**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.2.9 USBGetOneEnrollData”.**

3.2.10 FK_USBGetOneEnrollDataWithString

Type	**long FK_USBGetOneEnrollDataWithString(long nHandleIndex, long * pnEnrollNumber, long * pnBackupNumber, long * pnMachinePrivilege, LPSTR *apstrEnrollData, long * pnEnableFlag, LPSTR * dwEnrollName)**	
Functionality	**To get the enrollment data in the form of strings. It is equal to “FK_USBGetOneEnrollData”.**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**

	2	For the details, please refer to “0 USBGetOneEnrollDataWithString”.

3.2.11 FK_USBSetOneEnrollData

Type	long FK_USBSetOneEnrollData(long nHandleIndex, long nEnrollNumber, long nBackupNumber, long nMachinePrivilege, void * pnEnrollData, long nPassWord, long nEnableFlag, char *dwEnrollName)	
Functionality	To form, in the memory of the PC, the operational authorization and enrollment data of the person to be registered to turn them into a file usable in a USB memory	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.11 USBSetOneEnrollData”.

3.2.12 FK_USBSetOneEnrollDataWithString

Type	long FK_USBSetOneEnrollDataWithString(long nHandleIndex, long nEnrollNumber, long nBackupNumber, long nMachinePrivilege, char *apstrEnrollData, long nEnableFlag, char *dwEnrollName)	
Functionality	To contain the enrollment data in the form of strings. It is equal to “FK_USBSetOneEnrollData”.	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.12 USBSetOneEnrollDataWithString”.

3.2.13 FK_USBWriteAllEnrollDataToFile

Type	long FK_USBWriteAllEnrollDataToFile(long nHandleIndex, char * pstrFilePath)	
Functionality	To compose files of the enrollment data formed in the memory of the PC by “FK_USBSetOneEnrollData” or “FK_USBSetOneEnrollDataWithString”	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.13 USBWriteAllEnrollDataToFile”.

3.2.14 FK_ReadAllUserID

Type	long FK_ReadAllUserID(long nHandleIndex)	
Functionality	To read into the memory of the PC all the registrants-relating information registered on the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.14 ReadAllUserID”.

3.2.15 FK_GetAllUserID

Type	long FK_GetAllUserID(long nHandleIndex, long * pnEnrollNumber, long * pnBackupNumber, long * pnMachinePrivilege, long * pnEnableFlag)	
Functionality	To get, one by oe, the registrants-relating information read with FK_ReadAllUserID	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.15 GetAllUserID”.

3.2.16 FK_EmptyEnrollData

Type	long FK_EmptyEnrollData(long nHandleIndex)	
Functionality	To delete all the registered enrollment data from the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.16 EmptyEnrollData”.

3.2.17 FK_ClearKeeperData

Type	long FK_ClearKeeperData(long nHandleIndex)	
Functionality	To delete all of the enrollment data and recorded data from the device (it means initializing the device.)	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.17 ClearKeeperData”.

3.2.18 FK_BenumbAllManager

Type	long FK_BenumbAllManager(long nHandleIndex)	
Functionality	To delete all of the administrative authorization information from the enrollment data and set registrants to general users	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.18 BenumbAllManager”.

3.2.19 FK_GetVerifyMode

Type	long FK_GetVerifyMode(long nHandleIndex ,long anEnrollNumber, long *apnVerifyMode)	
Functionality	To get veify mode information relating to the users to set the registrants to general users	
Others	1	nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.19 GetVerifyMode”.

3.2.20 FK_SetVerifyMode

Type	long FK_SetVerifyMode(long nHandleIndex ,long anEnrollNumber, long anVerifyMode)	
Functionality	To set veify mode information relating to the users to set the registrants to general users	
Others	1	nHandleIndex” is a COM port`s ID number returned by FK_ConnectComm” or FK_ConnectNet”.
	2	For the details, please refer to “2.2.20 SetVerifyMode”.

3.2.21 FK_USBGetOneEnrollData_1

Type	long FK_USBGetOneEnrollData_1((long nHandleIndex ,long *apnEnrollNumber, long *apnBackupNumber, long *apnVerifyMode, long *apnMachinePrivilege, long *apnEnrollData, long *apnPassWord, long *apnEnableFlag, BSTR *apnEnrollName)	
Functionality	To get the enrollment data read with a command “USBReadAllEnrollDataFromFile”.	
Others	1	nHandleIndex” is a COM port`s ID number returned by FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.2.21 USBGetOneEnrollData_1”.

3.2.22 FK_USBGetOneEnrollDataWithString_1

Type	**long FK_USBGetOneEnrollDataWithString_1(long nHandleIndex ,long *apnEnrollNumber, long *apnBackupNumber, long *apnVerifyMode,long *apnMachinePrivilege, BSTR* apstrEnrollData, long *apnEnableFlag, BSTR *apnEnrollName)**	
Others	**1**	**nHandleIndex" is a COM port`s ID number returned by FK_ConnectComm" or "FK_ConnectNet".**
	2	**For the details, please refer to "2.2.22 USBGetOneEnrollDataWithString_1".**

3.2.23 FK_USBSetOneEnrollData_1

Type	**long USBSetOneEnrollData(long nHandleIndex ,long anEnrollNumber, long anBackupNumber, long anVerifyMode,long anMachinePrivilege, long *apnEnrollData, long anPassWord, long anEnableFlag, LPCTSTR anEnrollName)**	
Others	**1**	**nHandleIndex" is a COM port`s ID number returned by "FK_ConnectComm" or "FK_ConnectNet".**
	2	**For the details, please refer to "2.2.23 USBSetOneEnrollData_1".**

3.2.24 FK_USBSetOneEnrollDataWithString_1

Type	**long FK_USBSetOneEnrollDataWithString_1(long nHandleIndex ,long anEnrollNumber, long anBackupNumber,** **long anVerifyMode, long anMachinePrivilege, BSTR apstrEnrollData, long anEnableFlag, LPCTSTR anEnrollName)**	
Others	**1**	**nHandleIndex" is a COM port`s ID number returned by "FK_ConnectComm" or "FK_ConnectNet".**
	2	**For the details, please refer to "2.2.24 USBSetOneEnrollDataWithString_1".**

3.2.25 FK_USBReadAllEnrollDataFromFile_Color

Type	**long FK_USBReadAllEnrollDataFromFile_Color (long nHandleIndex, char * pstrFilePath)**	
Functionality	**To read the enrollment data from the relevant file formed in the USB memory into the internal memory of the PC and make an analysis of them**	
Others	**1**	**"nHandleIndex" is a COM port`s ID number returned by "FK_ConnectComm" or "FK_ConnectNet".**
	2	**For the details, please refer to "2.2.725 USBReadAllEnrollDataFromFile".**

3.2.26 FK_USBWriteAllEnrollDataToFile_Color

Type	**long FK_USBWriteAllEnrollDataToFile_Color (long nHandleIndex, char * pstrFilePath, long anNewsKind)**	
Functionality	**To compose files of the enrollment data formed in the memory of the PC by "FK_USBSetOneEnrollData" or "FK_USBSetOneEnrollDataWithString"**	
Others	**1**	**"nHandleIndex" is a COM port`s ID number returned by "FK_ConnectComm" or "FK_ConnectNet".**
	2	**For the details, please refer to "2.2.1326 USBWriteAllEnrollDataToFile_Color".**

3.2.27 FK_USBGetOneEnrollData_Color

Type	**long FK_USBGetOneEnrollData_Color (long nHandleIndex, long * pnEnrollNumber, long * pnBackupNumber, long * pnMachinePrivilege, void * pnEnrollData, long * pnPassWord, long * pnEnableFlag, LPSTR * dwEnrollName, long anNewsKind)**
Functionality	**To get the enrollment data read with command "USBReadAllEnrollDataFromFile"**

Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.2.927 USBGetOneEnrollData_Color”.**

3.2.28 FK_USBGetOneEnrollDataWithString_Color

Type	**long FK_USBGetOneEnrollDataWithString_Color (long nHandleIndex, long * pnEnrollNumber, long * pnBackupNumber, long * pnMachinePrivilege, LPSTR *apstrEnrollData, long * pnEnableFlag, LPSTR * dwEnrollName, long anNewsKind)**	
Functionality	**To get the enrollment data in the form of strings. It is equal to “FK_USBGetOneEnrollData_Color”.**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.2.28 USBGetOneEnrollDataWithString_Color”.**

3.2.29 FK_USBSetOneEnrollData_Color

Type	**long FK_USBSetOneEnrollData_Color(long nHandleIndex, long nEnrollNumber, long nBackupNumber, long nMachinePrivilege, void * pnEnrollData, long nPassWord, long nEnableFlag, char *dwEnrollName, long anNewsKind)**	
Functionality	**To form, in the memory of the PC, the operational authorization and enrollment data of the person to be registered to turn them into a file usable in a USB memory**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.2.1129 USBSetOneEnrollData_Color”.**

3.2.30 FK_USBSetOneEnrollDataWithString_Color

Type	**long FK_USBSetOneEnrollDataWithString_Color (long nHandleIndex, long nEnrollNumber, long nBackupNumber, long nMachinePrivilege, char *apstrEnrollData, long nEnableFlag, char *dwEnrollName, long anNewsKind)**	
Functionality	**To contain the enrollment data in the form of strings. It is equal to “FK_USBSetOneEnrollData”.**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.2.1230 USBSetOneEnrollDataWithString_Color”.**

3.3 Management of Recorded Data

3.3.1 FK_LoadSuperLogData

Type	**long FK_LoadSuperLogData(long nHandleIndex, long nReadMark)**	
Functionality	**To read the management data from the device into the memory of the PC and make an analysis of them**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.3.1 LoadSuperLogData”.**

3.3.2 FK_USBLoadSuperLogDataFromFile

Type	long FK_USBLoadSuperLogDataFromFile(long nHandleIndex, char *astrFilePath)	
Functionality	To read the recorded data from the relevant file formed in the USB memory into the memory of the PC and make an analysis of them	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.3.2 USBLoadSuperLogDataFromFile”.

3.3.3 FK_GetSuperLogData

Type	long FK_GetSuperLogData(long nHandleIndex, long *pnSEnrollNumber, long *pnGEnrollNumber, long *pnManipulation, long *pnBackupNumber, DATE *pnDateTime)	
Functionality	To get, one by oe, the management data read into the memory of the PC with “FK_LoadSuperLogData” or “FK_USBLoadSuperLogDataFromFile”	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “0 GetSuperLogData”.

3.3.4 FK_EmptySuperLogData

Type	long FK_EmptySuperLogData(long nHandleIndex)	
Functionality	To delete all the management data from the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.3.4 EmptySuperLogData”.

3.3.5 FK_LoadGeneralLogData

Type	long FK_LoadGeneralLogData(long nHandleIndex, long nReadMark)	
Functionality	To read the incoming and outgoing data from the device into the memory of the PC and make an analysis of them	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.3.5 LoadGeneralLogData”.

3.3.6 FK_USBLoadGeneralLogDataFromFile

Type	long FK_USBLoadGeneralLogDataFromFile(long nHandleIndex, char * pstrFilePath)	
Functionality	To read the incoming and outgoing data from the relevant file formed in the USB memory into the memory of the PC and make an analysis of them	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.3.6 USBLoadGeneralLogDataFromFile”.

3.3.7 FK_GetGeneralLogData

Type	long FK_GetGeneralLogData(long nHandleIndex, long * pnEnrollNumber, long *pnVerifyMode, long *pnInOutMode, DATE *pnDateTime)
Functionality	To get, one by one, the incoming and outgoing daa read in the memory of the PC with command “FK_LoadGeneralLogData” or “FK_USBLoadGeneralLogDataFromFile”

Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.3.7 GetGeneralLogData”.**

3.3.8 FK_EmptyGeneralLogData

Type	**long FK_EmptyGeneralLogData(long nHandleIndex)**	
Functionality	**To delete all the incoming and outgoing data from the device**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.3.8 EmptyGeneralLogData”.**

3.3.9 FK_GetGeneralLogData_1

Type	**FK_GetGeneralLogData_1(long nHandleIndex ,long *apnEnrollNumber, long pnVerifyMode , long *apnInOutMode, long *apnYear, long *apnMonth, long *apnDay, long *apnHour, long *apnMinute, long *apnSec)**	
Functionality	**To get, one by one, the attendance data read in the memory of the PC by a command “LoadGeneralLogData” “USBLoadGeneralLogDataFromFile”.**	
Parameter	**apnEnrollNumber**	**Variable pointer of the registration number of the registrant coming in or going out**
	apnVerifyMode	**Variable pointer of the verification mode**
	apnInOutMode	**Variable pointer of the mode of coming in or going out**
	apnYear,apnMonth apnDay, apnHour apnMinute, apnSec	**Variable pointer of the time and day when the registrant came in or went out**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.3.89 EmptyGeneralLogData_1”.**

3.3.10 FK_GetSuperLogData_1

Type	**long FK_GetSuperLogData_1(long nHandleIndex ,long *apnSEnrollNumber, long *apnGEnrollNumber, long *apnManipulation, long *apnBackupNumber, long *apnYear, long *apnMonth, long *apnDay, long *apnHour,long *apnMinute, long *apnSec)**	
Functionality	**To get, one by one, the management data read into the memory of the PC with a command “LoadSuperLogData” or “USBLoadSuperLogDataFromFile”.**	
Parameter	**apnSEnrollNumber**	**Variable pointer of the registration number of the manager**
	apnGEnrollNumber	**Variable pointer of the registration number of the managed**
	apnManipulation	**Variable pointer of the identification number of the managed**
	apnBackupNumber	**Variable pointer of the number classifying the kind of the enrollment data of the managed person**
	apnYear, apnMonth apnDay, apnHour apnMinute, apnSec	**Variable pointer of the time and the date when the management was recorded**
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”.**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**

	2	For the details, please refer to “2.3.810 EmptyGeneralLogData_1”.

3.3.11 FK_GetRealTimeInfo

Type	FK_GetRealTimeInfo(long* apGetRealTime)	
Functionality	To export to the PC the waiting time for transfer of blocks and sectors of time for automatic uploading of transactions	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.4.11 EnableUser”.

3.3.12 FK_SetRealTimeInfo

Type	FK_SetRealTimeInfo(long* apSetRealTime)	
Functionality	To write into machines the waiting time for transfer of blocks and sectors of time for automatic uploading of transactions。	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.4.1 EnableUser”.

3.4 Management of Registrant Information

3.4.1 FK_EnableUser

Type	long FK_EnableUser(long nHandleIndex, long nEnrollNumber, long nBackupNumber, long nEnableFlag)	
Functionality	To enable/forbid the registrant to use the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.4.1 EnableUser”.

3.4.2 FK_ModifyPrivilege

Type	long FK_ModifyPrivilege(long nHandleIndex, long nEnrollNumber, long nBackupNumber, long nMachinePrivilege)	
Functionality	To set the operational authorization of the registrant upon the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.4.2 ModifyPrivilege”.

3.4.3 FK_GetUserName

Type	long FK_GetUserName(long nHandleIndex, long nEnrollNumber, LPSTR *pstrUserName)	
Functionality	To get the name granted to the registrant	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.4.3 GetUserName”.

3.4.4 FK_SetUserName

Type	long FK_SetUserName(long nHandleIndex, long nEnrollNumber, char *pstrUserName)

Functionality	To assign a name to the registrant	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.4.4 SetUserName”.

3.4.5 FK_GetNewsMessage

Type	long FK_GetNewsMessage(long nHandleIndex, long nNewsId, LPSTR *pstrNews)	
Functionality	To get the designated message out from the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.4.5 GetNewsMessage”.

3.4.6 FK_SetNewsMessage

Type	long FK_SetNewsMessage(long nHandleIndex, long nNewsId, char * pstrNews)	
Functionality	To set a message into the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.4.6 SetNewsMessage”.

3.4.7 FK_GetUserNewsID

Type	long FK_GetUserNewsID(long nHandleIndex, long nEnrollNumber, long * pnNewsId)	
Functionality	To get the ID number of the message assigned to the registrant	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.4.7 GetUserNewsID”.

3.4.8 FK_SetUserNewsID

Type	long FK_SetUserNewsID(long nHandleIndex, long nEnrollNumber, long nNewsId)	
Functionality	To assign the registrant an ID number of the message	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.4.8 SetUserNewsID”.

3.5 Management of Device

3.5.1 FK_EnableDevice

Type	long FK_EnableDevice(long nHandleIndex, unsigned char nEnableFlag)	
Functionality	To allow or forbid the operations upon the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.5.1 EnableDevice”.

3.5.2 FK_PowerOnAllDevice

Type	void FK_PowerOnAllDevice(long nHandleIndex)

Functionality	To run the connected devices	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.5.2 PowerOnAllDevice”.

3.5.3 FK_PowerOffDevice

Type	long FK_PowerOffDevice(long nHandleIndex)	
Functionality	To power off the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.5.3 PowerOffDevice”.

3.5.4 FK_GetDeviceTime

Type	long FK_GetDeviceTime(long nHandleIndex, DATE * pnDateTime)	
Functionality	To get time and dates of the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.5.4 GetDeviceTime”.

3.5.5 FK_SetDeviceTime

Type	long FK_SetDeviceTime(long nHandleIndex, DATE nDateTime)	
Functionality	Set time and dates on the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.5.5 SetDeviceTime”.

3.5.6 FK_GetDeviceStatus

Type	long FK_GetDeviceStatus(long nHandleIndex, long nStatusIndex, long *pnValue)	
Functionality	To get the status values from the current device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.5.6 GetDeviceStatus”.

3.5.7 FK_GetDeviceInfo

Type	long FK_GetDeviceInfo(long nHandleIndex, long nInfoIndex, long *pnValue)	
Functionality	To get the information about the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.5.7 GetDeviceInfo”.

3.5.8 FK_SetDeviceInfo

Type	long FK_SetDeviceInfo(long nHandleIndex, long nInfoIndex, long nValue)	
Functionality	To set information into the device	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.5.8 SetDeviceInfo”.

3.5.9 FK_GetProductData

Type	long FK_GetProductData(long nHandleIndex, long nDataIndex, char *pstrValue)	
Functionality	To get the sales information entered by the seller	
Others	1	"nHandleIndex" is a COM port`s ID number returned by "FK_ConnectComm" or "FK_ConnectNet".
	2	For the details, please refer to "2.5.9 GetProductData".

3.5.10 FK_GetProductDataWithString

Type	long FK_GetProductDataWithString(long nHandleIndex, long nDataIndex, BSTR *apstrValue)	
Functionality	To get the sales information entered by the seller in the form of strings	
Others	1	"nHandleIndex" is a COM port`s ID number returned by "FK_ConnectComm" or "FK_ConnectNet".
	2	For the details, please refer to "2.5.9 GetProductData".

3.5.11 FK_GetDeviceVersion

Type	long FK_GetDeviceVersion(long nHandleIndex, long *pnVersion)	
Functionality	To get a version containing the revision history of every model	
Others	1	"nHandleIndex" is a COM port`s ID number returned by "FK_ConnectComm" or "FK_ConnectNet".
	2	For the details, please refer to "2.5.10 GetDeviceVersion".

3.5.12 FK_GetDeviceTime_1

Type	long FK_GetDeviceTime_1(long nHandleIndex, long *apnYear, long *apnMonth, long *apnDay, long apnHour, long apnMinute, long apnSec, long *apnDayOfWeek)	
Functionality	To get the time and date of the device	
Parameter	apnYear,apnMonth apnDay, apnHour apnMinute, apnSec apnDayOfWeek	Variable pointer of time and dates
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".	
Others	1	"nHandleIndex" is a COM port`s ID number returned by "FK_ConnectComm" or "FK_ConnectNet".
	2	For the details, please refer to "2.5.101 GetDeviceVersion".

3.5.13 FK_SetDeviceTime_1

Type	long SetDeviceTime_1(long nHandleIndex, long anYear, long anMonth, long anDay, long anHour, long anMinute, long anSec, long anDayOfWeek)	
Functionality	To set time and a date on the device	
Parameter	anYear, anMonth anDay, anHour anMinute, anSec anDayOfWeek	Time and date data
Return	Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".	

Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.5.102 GetDeviceVersion”.**

3.6 Management of Bells

3.6.1 FK_GetBellTime

Type	**long FK_GetBellTime(long nHandleIndex, long * pnBellCount, long * ptBellInfo)**	
Functionality	**To get setting information about the bell**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.6.1 GetBellTime”.**

3.6.2 FK_GetBellTimeWithString

Type	**long FK_GetBellTimeWithString(long nHandleIndex, long *pnBellCount, LPSTR *apstrBellInfo)**	
Functionality	**Equal to command “FK_GetBellTime”, it gets the bell information in the form of strings.**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.6.2 GetBellTimeWithString”.**

3.6.3 FK_SetBellTime

Type	**long FK_SetBellTime(long nHandleIndex, long nBellCount, long * ptBellInfo)**	
Functionality	**To set the information about the bell into the device**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.6.3 SetBellTime”.**

3.6.4 FK_SetBellTimeWithString

Type	**long FK_SetBellTimeWithString(long nHandleIndex, long nBellCount, char *apstrBellInfo)**	
Functionality	**Equal to command “FK_SetBellTime”, it set the bell information in the form of strings**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.6.4 SetBellTimeWithString”.**

3.7 Control of Doors

This function is not supported for some models.

3.7.1 FK_GetDoorStatus

Type	**long FK_GetDoorStatus(long nHandleIndex, long *apnStatusVal)**	
Functionalityt	**To get the door opening status**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**

	2	For the details, please refer to “2.7.1 GetDoorStatus”.

3.7.2 FK_SetDoorStatus

Type	long FK_SetDoorStatus(long nHandleIndex, long anStatusVal)	
Functionality	To control the door opening status	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.7.2 SetDoorStatus”.

3.7.3 FK_GetPassTime

Type	long FK_GetPassTime(long nHandleIndex, long anPassTimeID, long *apnPassTime, long anPassTimeSize)	
Functionality	To get the time zone information for opening or closing the door	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.7.3 GetPassTime”.

3.7.4 FK_GetPassTimeWithString

Type	long FK_GetPassTimeWithString(long nHandleIndex, long anPassTimeID, LPSTR *apstrPassTime)	
Functionality	Equal to command “FK_GetPassTime”, it returns the time zone information in the form of strings.	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.7.4 GetPassTimeWithString”.

3.7.5 FK_SetPassTime

Type	long FK_SetPassTime(long nHandleIndex, long anPassTimeID, long *apnPassTime, long anPassTimeSize)	
Functionality	To set time zone information about opening or closing the door	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.7.5 SetPassTime”.

3.7.6 FK_SetPassTimeWithString

Type	long FK_SetPassTimeWithString(long nHandleIndex, long anPassTimeID, char *apstrPassTime)	
Functionality	Equal to command “FK_SetPassTime”, it contains the time zone information in the form of strings.	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.7.6 SetPassTimeWithString”.

3.7.7 FK_GetUserPassTime

Type	long FK_GetUserPassTime(long nHandleIndex, long anEnrollNumber, long *apnGroupID, long *apnPassTimeID, long anPassTimeIDSize)

Functionality	**To get the group of the time zone information assigned to the designated registrant and those assigned individually**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.7.7 GetUserPassTime”.**

3.7.8 FK_GetUserPassTimeWithString

Type	**long FK_GetUserPassTimeWithString(long nHandleIndex, long anEnrollNumber, long *apnGroupID, LPSTR *apstrPassTimeID)**	
Functionality	**Equal to command “FK_GetUserPassTime”, it returns the structure of the ID numbers in the form of strings.**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.7.8 GetUserPassTimeWithString”.**

3.7.9 FK_SetUserPassTime

Type	**long FK_SetUserPassTime(long nHandleIndex, long anEnrollNumber, long anGroupID, long *apnPassTimeID, long anPassTimeIDSize)**	
Functionality	**To set the group of the time zone information assigned to the designated registrant and those assigned individually**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the detail, please refer to “2.7.9 SetUserPassTime”.**

3.7.10 FK_SetUserPassTimeWithString

Type	**long FK_SetUserPassTimeWithString(long nHandleIndex, long anEnrollNumber, long anGroupID, char *apstrPassTimeID)**	
Functionality	**Equal to command “FK_SetUserPassTime”, it contains the structure of ID numbers in the form of strings.**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.7.10 SetUserPassTimeWithString”.**

3.7.11 FK_GetGroupPassTime

Type	**long FK_GetGroupPassTime(long nHandleIndex, long anGroupID, long *apnPassTimeID, long anPassTimeIDSize)**	
Functionality	**To get the ID numbers of the time zone information corresponding to the designated group of the information**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.7.11 GetGroupPassTime”.**

3.7.12 FK_GetGroupPassTimeWithString

Type	**long FK_GetGroupPassTimeWithString(long nHandleIndex, long anGroupID, LPSTR *apstrPassTimeID)**	
Functionality	**Equal to command “FK_GetGroupPassTime”, it returns the structure of ID numbers in the form of strings.**	
Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**

	2	For the details, please refer to “2.7.12 GetGroupPassTimeWithString”.

3.7.13 FK_SetGroupPassTime

Type	long FK_SetGroupPassTime(long nHandleIndex, long anGroupID, long *apnPassTimeID, long anPassTimeIDSize)	
Functionality	To set ID numbers of the time zone information in the designated group	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.7.13 SetGroupPassTime”.

3.7.14 FK_SetGroupPassTimeWithString

Type	long FK_SetGroupPassTimeWithString(long nHandleIndex, long anGroupID, char *apstrPassTimeID)	
Functionality	Equal to command “FK_SetGroupPassTime”, it contains the structure of ID numbers in the form of strings.	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.7.14 SetGroupPassTimeWithString”.

3.7.15 FK_GetGroupMatch

Type	long FK_GetGroupMatch(long nHandleIndex, long *apnGroupMatch, long anGroupMatchSize)	
Functionality	To get the door control union of the group of the time zone information structure	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.7.15 GetGroupMatch”.

3.7.16 FK_GetGroupMatchWithString

Type	long FK_GetGroupMatchWithString(long nHandleIndex, LPSTR *apstrGroupMatch)	
Functionality	Equal to command “FK_GetGroupMatchTime”, it returns the structure of the union in the form of strings.	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.7.16 GetGroupMatchWithString”.

3.7.17 FK_SetGroupMatch

Type	long FK_SetGroupMatch(long nHandleIndex, long *apnGroupMatch, long anGroupMatchSize)	
Functionality	To set the door control union of the group of the time zone information structure	
Others	1	“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.
	2	For the details, please refer to “2.7.17 SetGroupMatch”.

3.7.18 FK_SetGroupMatchWithString

Type	long FK_SetGroupMatchWithString(long nHandleIndex, char *apstrGroupMatch)
Functionality	Equal to command “FK_SetGroupMatch”, it contains the structure of the union in the form of strings.

Others	**1**	**“nHandleIndex” is a COM port`s ID number returned by “FK_ConnectComm” or “FK_ConnectNet”.**
	2	**For the details, please refer to “2.7.18 SetGroupMatchWithString”.**

3.8 Adjust Mangement

This function is not supported for some models.

3.8.1 FK_GetAdjustInfo

Type	**long GetAdjustInfo(long nHandleIndex, long* dwAdjustedState, long* dwAdjustedMonth, long* dwAdjustedDay, long* dwAdjustedHour, long* dwAdjustedMinute, long* dwRestoredState, long* dwRestoredMonth, long* dwRestoredDay, long* dwRestoredHour, long* dwRestoredMinute)**	
Functionality	**To get a daylight saving time**	
others	**1**	**For details, please refer to 《4.1.6 ADJUSTINFO Structure》.**

3.8.2 FK_SetAdjustInfo

Type	**long SetAdjustInfo(long nHandleIndex ,long dwAdjustedState, long dwAdjustedMonth, long dwAdjustedDay, long dwAdjustedHour, long dwAdjustedMinute, long dwRestoredState, long dwRestoredMonth, long dwRestoredDay, long dwRestoredHour, long dwRestoredMinute)**	
Functionality	**To set a daylight saving time**	
others	**1**	**For details, please refer to 《4.1.6 ADJUSTINFO Structure》.**

3.9 Network Information Management

3.9.1 GetServerNetInfo

Type	**long FK_GetServerNetInfo(long nHandleIndex BSTR* astrServerIPAddress, long* apServerPort, long* apServerRequest)**	
Functionality	**To get a server information**	
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
others	**1**	**"nHandleIndex" is a COM port`s ID number returned by "FK_ConnectComm" or "FK_ConnectNet".**
	2	**For the details, please refer to "2.7.16 GetGroupMatchWithString".**

3.9.2 SetServerNetInfo

Type	**long FK_SetServerNetInfo(long nHandleIndex , LPCTSTR astrServerIPAddress, long anServerPort, long anServerRequest)**	
Functionality	**To set server informationa**	
Return	**Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table".**	
others	**1**	**"nHandleIndex" is a COM port`s ID number returned by "FK_ConnectComm" or "FK_ConnectNet".**
	2	**For the details, please refer to "2.7.162 GetGroupMatchWithString".**

3.9.3 SetUSBModel

Type	**void FK_SetUSBModel(long anModel)**
Functionality	**To set machine model info for USB Flash information**

others	1	“anModel” is a machine model info. #define FK625_FP1000 2001 #define FK625_FP2000 2002 #define FK625_FP3000 2003 #define FK625_FP5000 2004 #define FK625_FP10000 2005 #define FK625_FP30000 2006 #define FK625_ID30000 2007 #define FK635_FP700 3001 #define FK635_FP3000 3002 #define FK635_FP10000 3003 #define FK635_ID30000 3004 #define FK723_FP1000 4001 #define FK725_FP1000 5001 #define FK725_FP1500 5002 #define FK725_ID5000 5003 #define FK725_ID30000 5004 #define FK735_FP500 6001 #define FK735_FP3000 6002 #define FK735_ID30000 6003 #define FK925_FP3000 7001 #define FK935_FP3000 8001.

4 Appendix

4.1 Structures

4.1.1 BELLINFO Structure

```
#define MAX_BELLCOUNT_DAY          24
#define MAX_BELLCOUNT_WEEK         7
#define    BELLKIND_NONE           0
#define    BELLKIND_BUZZER         1
#define    BELLKIND_BELL           2
#define    BELLKIND_BUZZERBELL     3

/*--- Bell Time Infomation ---*/
typedef struct tagBELLTIMEINFO {
    BYTE        Mark;                       // Setting Mark
    BYTE        WeekDay;                    // Day
    BYTE        Reserve[2];                 // Reserve
    BYTE        Valid[MAX_BELLCOUNT_DAY];   // Flag for valid setting of bells
    BYTE        Hour[MAX_BELLCOUNT_DAY];    // Time of bells ringing (hour)
    BYTE        Minute[MAX_BELLCOUNT_DAY];  // Time of bells ringing (minute)
    BYTE        BellKind[MAX_BELLCOUNT_DAY]; // Kind of bells ringing
} BELLTIMEINFO;
typedef struct tagBELLINFO {
    BYTE            BellHoldTime;
    BYTE            Reserve[3];
    BELLTIMEINFO    BellTime[MAX_BELLCOUNT_WEEK];
} BELLINFO;
```

4.1.2 PASSCTRLTIME Structure

```
#define MAX_PASSCTRLGROUP_COUNT   50
#define MAX_PASSCTRL_COUNT        7 // Pass Count Max Value
typedef struct tagPASSTIME {
    BYTE    StartHour;      // Time of opening doors (hour)
    BYTE    StartMinute;    // Time of opening doors (minute)
    BYTE    EndHour;        // Time of closing doors (hour)
    BYTE    EndMinute;      // Time of closing doors (minute)
} PASSTIME; // Information about time zone – a day
typedef struct tagPASSCTRLTIME {
```

PASSTIME mPassCtrlTime[MAX_PASSCTRL_COUNT]; // Information about time zone – every weekday

} PASSCTRLTIME; // Information about time zone – a week

4.1.3 USERPASSINFO Structure

```
#define MAX_USERPASSINFO_COUNT 3
typedef struct tagUSERPASSINFO {
    BYTE UserPassID[MAX_USERPASSINFO_COUNT]; // ID number of time zone information
} USERPASSINFO; // ID number of time zone information set onto the registrant
```

4.1.4 GROUPPASSINFO Structure

```
#define MAX_GROUPPASSKIND_COUNT 5
#define MAX_GROUPPASSINFO_COUNT 3
typedef struct tagGROUPPASSINFO {
    BYTE GroupPassID[MAX_GROUPPASSINFO_COUNT]; // ID number of time zone information
} GROUPPASSINFO; // Group of time zone information
```

4.1.5 GROUPMATCHINFO Structure

```
#define MAX_GROUPMATCHINFO_COUNT 10
typedef struct tagGroupMatchInfo {
    BYTE GroupMatch[MAX_GROUPMATCHINFO_COUNT];// ID number of group of time zone information
} GROUPMATCHINFO; // Union of groups of time zone information
```

4.1.6 ADJUSTNFO Structure

```
  typedef struct tagCHANGE_DATE {
    BYTE Month;  // Month
    BYTE Day;    // Day
    BYTE Hour;   // Hour
    BYTE Minute; // Minute
  } CHANGEDATE;
typedef struct tagADJUSTINFO {
   unsigned char  AdjustedState;  // Changed state
```

```
    unsigned char  Reserve1[1];       // Reserve
    unsigned short AdjustedFlag;      // Changed Flag
    CHANGEDATE     Adjusted;    // changed data
    unsigned char  RestoredState;    // Restored state
    unsigned char  Reserve2[1];      // Reserve
    unsigned short RestoredFlag;    // Restored flag
    CHANGEDATE     Restored;    // Restored data
} ADJUSTINFO;
```

4.1.7 REALTIMEINFO 结构体

```
#define MAX_REAL_TIME   4
typedef struct tagGroupMatchInfo {
    BYTE   Valid;     // senddong mode
    BYTE   AckTime; // acking time
    BYTE   WaitTIme;// wait time
    BYTE   Reserve; // reserve
    BYTE   SendPos; // Sending position
    BYTE   Hour[MAX_REAL_TIME]; // Hour of the TimeZone
    BYTE   Minute[MAX_REAL_TIME]; // Minute of the TimeZone
} REALTIMEINFO; // A structured body for setting waiting time for transfer of blocks and sectors of time for automatic uploading of transactions
```

4.2 Error Code Table

Value	Symbol	Description
1	RUN_SUCCESS	Message informing of the successful execution of commands
0	RUNERR_NOSUPPORT	Error that the device does not support the relevant command
-1	RUNERR_UNKNOWNERROR	Unknown error
-2	RUNERR_NO_OPEN_COMM	Error that the device has been not connected to
-3	RUNERR_WRITE_FAIL	Error that the data has not been transmitted to the device
-4	RUNERR_READ_FAIL	Error that the data has not been read from the device
-5	RUNERR_INVALID_PARAM	Error that the input parameters are not correct
-6	RUNERR_NON_CARRYOUT	Error that the command has not been executed correctly

-7	RUNERR_DATAARRAY_END	Message telling that there is no more data to get
-8	RUNERR_DATAARRAY_NONE	Error that the data do not exist
-9	RUNERR_MEMORY	Error that the memory of the PC is not enough
-10	RUNERR_MIS_PASSWORD	Error that the input license does not accord when connecting with the device
-11	RUNERR_MEMORYOVER	Error that the memory has no space where more enrollment data can be registered in the device
-12	RUNERR_DATADOUBLE	Error that the registration number to be enrolled is already stored in the database of the device
-14	RUNERR_MANAGEROVER	Error that the memory has no space where more data of the manager can be registered in the device
-15	RUNERR_FPDATAVERSION	Error that the version of the fingerprint data to be used is not correct