**Attendance/Access Control**

**CS SDK Manual**

**DATE:** **September,2019**

**Statement:**

* Please refer to the manual in detail. Hysoon have no responsibility for data loss caused by users or programs.
* This manual is designed on the basis of SDK 2.0.

Content

**[Attendance/Access Control](#_Toc22643919)** [- 1 -](#_Toc22643919)

[**CS SDK Manual** - 1 -](#_Toc22643920)

[**Statement:** 2](#_Toc22643921)

[Content 3](#_Toc22643922)

[1 .SDK Description 9](#_Toc22643923)

[2 .FK623Attend.DLL Interface 11](#_Toc22643924)

[2.1 Connection and Disconnection of Devices 11](#_Toc22643925)

[2.1.1 \* ConnectComm 11](#_Toc22643926)

[2.1.2 ConnectNet 12](#_Toc22643927)

[2.1.3 ConnectUSB 12](#_Toc22643928)

[2.1.4 DisConnect 13](#_Toc22643929)

[2.2 Management of Registered Data 13](#_Toc22643930)

[2.2.1 GetEnrollData 13](#_Toc22643931)

[2.2.2 GetEnrollData\_StringID 14](#_Toc22643932)

[2.2.3 PutEnrollData 15](#_Toc22643933)

[2.2.4 PutEnrollData\_StringID 16](#_Toc22643934)

[2.2.5 SaveEnrollData 16](#_Toc22643935)

[2.2.6 DeleteEnrollData 17](#_Toc22643936)

[2.2.7 USBReadAllEnrollDataFromFile 17](#_Toc22643937)

[2.2.8 \*USBReadAllEnrollDataCount 17](#_Toc22643938)

[2.2.9 USBGetOneEnrollData 18](#_Toc22643939)

[2.2.10 USBGetOneEnrollData\_StringID 19](#_Toc22643940)

[2.2.11 USBSetOneEnrollData 19](#_Toc22643941)

[2.2.12 USBSetOneEnrollData\_StringID 20](#_Toc22643942)

[2.2.13 USBWriteAllEnrollDataToFile 21](#_Toc22643943)

[2.2.14 ReadAllUserID 21](#_Toc22643944)

[2.2.15 GetAllUserID 22](#_Toc22643945)

[2.2.16 GetAllUserID\_StringID 22](#_Toc22643946)

[2.2.17 EmptyEnrollData 23](#_Toc22643947)

[2.2.18 ClearKeeperData 23](#_Toc22643948)

[2.2.19 BenumbAllManager 24](#_Toc22643949)

[2.2.20 \*GetVerifyMode 24](#_Toc22643950)

[2.2.21 \*SetVerifyMode 24](#_Toc22643951)

[2.2.22 \*USBGetOneEnrollData\_1 24](#_Toc22643952)

[2.2.23 \*USBGetOneEnrollDataWithString\_1 25](#_Toc22643953)

[2.2.24 \*USBSetOneEnrollData\_1 26](#_Toc22643954)

[2.2.25 \*USBSetOneEnrollDataWithString\_1 27](#_Toc22643955)

[2.2.26 \*USBReadAllEnrollDataFromFile\_Color 28](#_Toc22643956)

[2.2.27 \*USBWriteAllEnrollDataToFile\_Color 28](#_Toc22643957)

[2.2.28 \*USBGetOneEnrollData\_Color 29](#_Toc22643958)

[2.2.29 \*USBGetOneEnrollDataWithString\_Color 30](#_Toc22643959)

[2.2.30 \*USBSetOneEnrollData\_Color 31](#_Toc22643960)

[2.2.31 \*USBSetOneEnrollDataWithString\_Color 31](#_Toc22643961)

[2.3 Management of Recorded Data 32](#_Toc22643962)

[2.3.1 \*LoadSuperLogData 32](#_Toc22643963)

[2.3.2 \*USBLoadSuperLogDataFromFile 33](#_Toc22643964)

[2.3.3 \*GetSuperLogData 33](#_Toc22643965)

[2.3.4 \*EmptySuperLogData 34](#_Toc22643966)

[2.3.5 LoadGeneralLogData 35](#_Toc22643967)

[2.3.6 USBLoadGeneralLogDataFromFile 35](#_Toc22643968)

[2.3.7 GetGeneralLogData 36](#_Toc22643969)

[2.3.8 EmptyGeneralLogData 38](#_Toc22643970)

[2.3.9 \* GetGeneralLogData\_1 38](#_Toc22643971)

[2.3.10 \* GetSuperLogData\_1 40](#_Toc22643972)

[2.3.11 \*GetRealTimeInfo 42](#_Toc22643973)

[2.3.12 \*SetRealTimeInfo 42](#_Toc22643974)

[2.3.13 LoadGeneralLogDataByDate 42](#_Toc22643975)

[2.4 Management of Devices 43](#_Toc22643976)

[2.4.1 \*EnableDevice 43](#_Toc22643977)

[2.4.2 \*PowerOnAllDevice 43](#_Toc22643978)

[2.4.3 \*PowerOffDevice 43](#_Toc22643979)

[2.4.4 GetDeviceTime 43](#_Toc22643980)

[2.4.5 SetDeviceTime 44](#_Toc22643981)

[2.4.6 GetDeviceStatus 44](#_Toc22643982)

[2.4.7 GetDeviceInfo 45](#_Toc22643983)

[2.4.8 SetDeviceInfo 46](#_Toc22643984)

[2.4.9 GetProductData 46](#_Toc22643985)

[2.4.10 \*GetDeviceVersion 47](#_Toc22643986)

[2.4.11 \*GetDeviceTime\_1 47](#_Toc22643987)

[2.4.12 \*SetDeviceTime\_1 47](#_Toc22643988)

[2.5 Management of Registrants` Information 48](#_Toc22643989)

[2.5.1 \*EnableUser 48](#_Toc22643990)

[2.5.2 ModifyPrivilege 48](#_Toc22643991)

[2.5.3 GetUserName 49](#_Toc22643992)

[2.5.4 SetUserName 49](#_Toc22643993)

[2.5.5 \*GetNewsMessage 50](#_Toc22643994)

[2.5.6 \*SetNewsMessage 50](#_Toc22643995)

[2.5.7 \*GetUserNewsID 50](#_Toc22643996)

[2.5.8 \*SetUserNewsID 51](#_Toc22643997)

[2.6 Management of Bells 51](#_Toc22643998)

[2.6.1 GetBellTime 51](#_Toc22643999)

[2.6.2 \*GetBellTimeWithString 51](#_Toc22644000)

[2.6.3 SetBellTime 52](#_Toc22644001)

[2.6.4 \*SetBellTimeWithString 52](#_Toc22644002)

[2.7 \*Post & Shift Management 52](#_Toc22644003)

[2.7.1 GetOneShiftInfo 52](#_Toc22644004)

[2.7.2 SetOneShiftInfo 53](#_Toc22644005)

[2.7.3 GetOnePostInfo 54](#_Toc22644006)

[2.7.4 SetOnePostInfo 54](#_Toc22644007)

[2.7.5 GetUserInfo 55](#_Toc22644008)

[2.7.6 SetUserInfo 56](#_Toc22644009)

[2.7.7 GetPostShiftInfo 56](#_Toc22644010)

[2.7.8 SetPostShiftInfo 57](#_Toc22644011)

[2.7.9 GetUserInfoEx 57](#_Toc22644012)

[2.7.10 SetUserInfoEx 57](#_Toc22644013)

[2.8 Photo Management 58](#_Toc22644014)

[2.8.1 GetEnrollPhoto 58](#_Toc22644015)

[2.8.2 SetEnrollPhoto 58](#_Toc22644016)

[2.8.3 DeletetEnrollPhoto 58](#_Toc22644017)

[2.9 Control of Doors 59](#_Toc22644018)

[2.9.1 GetDoorStatus 59](#_Toc22644019)

[2.9.2 SetDoorStatus 59](#_Toc22644020)

[2.9.3 GetTimeZone 59](#_Toc22644021)

[2.9.4 SetTimeZone 60](#_Toc22644022)

[2.9.5 GetUserWeekPassTime 60](#_Toc22644023)

[2.9.6 SetUserWeekPassTime 60](#_Toc22644024)

[2.9.7 FK\_ExtCommand 61](#_Toc22644025)

[2.9.8 \*GetPassTime 61](#_Toc22644026)

[2.9.9 \*GetPassTimeWithString 61](#_Toc22644027)

[2.9.10 \*SetPassTime 62](#_Toc22644028)

[2.9.11 \*SetPassTimeWithString 62](#_Toc22644029)

[2.9.12 \*GetUserPassTime 62](#_Toc22644030)

[2.9.13 \*GetUserPassTimeWithString 63](#_Toc22644031)

[2.9.14 \*SetUserPassTime 64](#_Toc22644032)

[2.9.15 \*SetUserPassTimeWithString 64](#_Toc22644033)

[2.9.16 \*GetGroupPassTime 65](#_Toc22644034)

[2.9.17 \*GetGroupPassTimeWithString 65](#_Toc22644035)

[2.9.18 \*SetGroupPassTime 66](#_Toc22644036)

[2.9.19 \*SetGroupPassTimeWithString 66](#_Toc22644037)

[2.9.20 \*GetGroupMatch 67](#_Toc22644038)

[2.9.21 \*GetGroupMatchWithString 67](#_Toc22644039)

[2.9.22 \*SetGroupMatch 68](#_Toc22644040)

[2.9.23 \*SetGroupMatchWithString 68](#_Toc22644041)

[2.10 \*Summer time management (deleted) 68](#_Toc22644042)

[2.10.1 GetAdjustInfo 68](#_Toc22644043)

[2.10.2 SetAdjustInfo 69](#_Toc22644044)

[2.11 \* Equipment information management 69](#_Toc22644045)

[2.11.1 GetServerNetInfo 69](#_Toc22644046)

[2.11.2 SetServerNetInfo 70](#_Toc22644047)

[2.11.3 SetUSBModel 70](#_Toc22644048)

[2.12 Card/Password(Refer to CardPwdReference.cs) 70](#_Toc22644049)

[2.12.1 string card to byte array 70](#_Toc22644050)

[2.12.2 string password to byte array 70](#_Toc22644051)

[2.12.3 Show card and password(refer to function of below) 72](#_Toc22644052)

[2.13 The Common Interface of JsonCommand 72](#_Toc22644053)

[*2.13.1 Import function* 73](#_Toc22644054)

[2.13.2 Enter Enroll 73](#_Toc22644055)

[2.13.3 Set Lock Control 74](#_Toc22644056)

[2.13.4 Set Attend Status 75](#_Toc22644057)

[2.13.5 Get Lock Control Params 75](#_Toc22644058)

[2.13.6 Get Attend Status 76](#_Toc22644059)

[2.13.7 JsonCommand using attentions 76](#_Toc22644060)

[2.14 Notes on use of DLL interface 76](#_Toc22644061)

[3 RealSvrOcxTcp.ocx Component 79](#_Toc22644062)

[3.1 Introduction to the use of Demo: 79](#_Toc22644063)

[3.2 Introduction to control usage 79](#_Toc22644064)

[4 Appendix 83](#_Toc22644065)

[4.1 Structures 83](#_Toc22644066)

[4.1.1 BELLINFO Structure 83](#_Toc22644067)

[4.1.2 PASSCTRLTIME Structure 83](#_Toc22644068)

[4.1.3 USERPASSINFO Structure 84](#_Toc22644069)

[4.1.4 GROUPPASSINFO Structure 84](#_Toc22644070)

[4.1.5 GROUPMATCHINFO Structure 84](#_Toc22644071)

[4.1.6 ADJUSTNFO Structure 84](#_Toc22644072)

[4.1.7 REALTIMEINFO Structure 85](#_Toc22644073)

[4.1.8 SetUSBModel Constants 85](#_Toc22644074)

[4.2 Error Code Table 86](#_Toc22644075)

[5 Version 87](#_Toc22644076)

[6.FAQs 88](#_Toc22644077)

[6.1 How to Obtain All Information of All Users? 88](#_Toc22644078)

[6.2 How to show Card/Passwords in the form of strings? 88](#_Toc22644079)

[6.3 How to push User Card/Password to Machine? 88](#_Toc22644080)

# .SDK Description

This manual describes an OEM program product FK623Attend which provides interfaces for development of applications using FKxxx series fingerprint time attendance terminals .

FK623Attend consists of, FK623Attend.dll FKAttend.dll and FKViaDev.dll for development of programs.

FK623Attend.dll is an interface DLL for connection of the devices with the applications.

FKAttend. DLL is a concrete implementation interface, for FK623Attend. DLL call

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

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

**Samples:**

Running FK623AttendDllCSSample.exe and FKRealSvrOcxTcpCSSample.exe test device interface in Execute&Dll folder. The device should be in the same LAN as the computer . device port defaults to 5005. device serve port defaults to 7005 when running FKRealSvrOcxTcpCSSample.exe to test

Real\_time monitoring function. Because the host port of the FKRealSvrOcxTcpCSSample.exe defaults to 7005.

Details of developing code can refer to Samples folder. You can use visual studio to open the interior code file for reference.

# .FK623Attend.DLL Interface

Note: the sections in \* are additional functions or has been abandoned. Please refer to the actual product

## Connection and Disconnection of Devices

### \* ConnectComm

|  |  |  |
| --- | --- | --- |
| Type | long ConnectComm(  long anMachineNumber,  long anComPort,  long anBaudRate,  BSTR astrTelNumber,  long anWaitDialTime,  long anLicense) | |
| Functionality | To open the COM port to connect to the device via the RS-232/485 cable. | |
| Parameter | anMachineNumber | Number granted to the device to be connected with |
| anComPort | Sequence number of COM port |
| anBaudRate | Communication baudrate |
| astrTelNumber | Telephone number |
| anWaitDialTime | Standby time for phone connection (the unit is ms.) |
| anLicense | License for connection |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 “astrTelNumber” and “anWaitDialTime” 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. | |

### ConnectNet

|  |  |  |
| --- | --- | --- |
| Type | long ConnectNet(  long anMachineNumber,  BSTR astrIpAddress,  long anPort,  long anTimeOut,  long anProtocolType,  long anNetPassword,  long anLicense) | |
| Functionality | To open the network port to connect with the device via the network cable. | |
| Parameter | anMachineNumber | Number granted to the device to be connected with |
| astrIpAddress | TCP/IP address of the device to be connected with |
| anPort | Sequence number of network port |
| anTimeOut | Standby time for the connection (the unit is ms.) |
| anProtocolType | Kind of protocol |
| anNetPassword | Network password |
| anLicense | License for connection |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “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.   1. : PROTOCOL\_TCPIP - TCP/IP communication 2. : PROTOCOL\_UDP - UDP communication （Deprecated）   3 “nLicense” has the same meaning as “ConnectComm”. | |

### ConnectUSB

|  |  |  |
| --- | --- | --- |
| Type | long ConnectUSB(  long anMachineNumber,  long anLicense) | |
| Functionality | To open the USB port to connect with the device via the USB cable. | |
| Parameter | anMachineNumber | Number granted to the device to be connected with |
| anLicense | License for connection |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 “nLicense” has the same meaning as “0 ConnectComm”. | |

### DisConnect

|  |  |  |
| --- | --- | --- |
| Type | void DisConnect() | |
| 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 | |

## Management of Registered Data

### 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 “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 “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 meaning of "anBackupNumber" is as follows:  0: BACKUP\_FP\_0 - represents the first fingerprint registered  ......  9: BACKUP\_FP\_9 - represents the ninth fingerprint registered  10: BACKUP\_PSW - represents the password registered  11: BACKUP\_CARD - represents the card registered  12: BACKUP\_FACE - represents a registered face  13:BACKUP\_PALMVEIN\_0 - represents the first palm print registered  14:BACKUP\_PALMVEIN\_1 - represents the second palm print registered | |

### GetEnrollData\_StringID

|  |  |  |
| --- | --- | --- |
| Type | public static extern int FK\_GetEnrollData\_StringID(int anHandleIndex, string apEnrollNumber, int anBackupNumber, ref int apnMachinePrivilege, byte[] apEnrollData, ref int apnPassWord); | |
| Functionality | To get the enrollment data in the type of a string. It is equal to GetEnrollData. | |
| Parameter | anHandleIndex |  |
| apEnrollNumber | 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 “Error Code Table”. | |
| Others | 1 The enrollment data is alwayse returned to “apstrEnrollData”  2 For other parameters, please refer to “GetEnrollData”. | |

### PutEnrollData

|  |  |  |
| --- | --- | --- |
| Type | long PutEnrollData(  long anEnrollNumber,  long anBackupNumber,  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 “Error Code Table”. | |
| Others | 1 As for “anBackupNumber”, please refer to “GetEnrollData”.  2 As for “anMachinePrivilege”, please refer to “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 “SaveEnrollData”. | |
|  |

### PutEnrollData\_StringID

|  |  |  |
| --- | --- | --- |
| Type | public static extern int FK\_PutEnrollData\_StringID(int anHandleIndex, string apEnrollNumber, int anBackupNumber, int anMachinePrivilege, byte[] apEnrollData, int anPassWord); | |
| Functionality | To contain the enrollment data in the type of a string. It is equal to PutEnrollData. | |
| Parameter | anHandleIndex, |  |
| apEnrollNumber | Registration number |
| anBackupNumber | Number classifying the kind of the enrollment data |
| anMachinePrivilege | Operational authorization of the registrants |
| apEnrollData | Variable pointer of the enrollment data |
|  | anPassWord |  |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 The enrollment data are alwayse are contained by “apstrEnrollData”.  2 As for the other parameters, please refer to “PutEnrollData”. | |

### SaveEnrollData

|  |  |  |
| --- | --- | --- |
| Type | long SaveEnrollData() | |
| 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 “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”. | |

### 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 “Error Code Table”. | |
| Others | 1 The command fails to be executed if the enrollment data do not exist in the device. | |

### USBReadAllEnrollDataFromFile

|  |  |  |
| --- | --- | --- |
| Type | long USBReadAllEnrollDataFromFile(BSTR astrFilePath) | |
| 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 | astrFilePath | File name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “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. | |

### \*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 “Error Code Table”. | |
| Others | 1 You should first read the data out with a command USBReadAllEnrollDataFromFile” before executing this command. | |

### USBGetOneEnrollData

|  |  |  |
| --- | --- | --- |
| Type | long USBGetOneEnrollData(  long\* apnEnrollNumber,  long\* apnBackupNumber,  long\* apnMachinePrivilege,  long\* apnEnrollData,  long\* apnPassWord,  long\* apnEnableFlag,  BSTR\* apstrEnrollName) | |
| Functionality | Read a single access through "USBReadAllEnrollDataFromFile" command to the registration data | |
| Parameter | apnEnrollNumber | Registration number |
| apnBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| apnMachinePrivilege | Variable pointer to the authorization of the registrants |
| apnEnrollData | Variable pointer to the enrolled data |
| apnPassWord | Variable pointer of data relating to password or cards |
| apnEnableFlag | User enabled flags |
| apstrEnrollName | User Name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 This instruction and "GetEnrollData" instruction similar and different parts are not directly connected to the device, applicable to the U disk.For a description of the "PutEnrollData" directive, refer to "PutEnrollData"  2 returns the code "RUNERR\_LOG\_END" after all the data is obtained  3. If no command "USBReadAllEnrollDataFromFile" read data to the PC, then this command will not be able to perform  Please refer to "EnableUser" for the meaning of apnEnableFlag. | |

### USBGetOneEnrollData\_StringID

|  |  |  |
| --- | --- | --- |
| Type | public static extern int FK\_USBGetOneEnrollData\_StringID(int anHandleIndex, [MarshalAs(UnmanagedType.LPStr)] ref string apEnrollNumber, ref int apnBackupNumber, ref int apnMachinePrivilege, byte[] apEnrollData, ref int apnPassWord, ref int apnEnableFlag, [MarshalAs(UnmanagedType.LPStr)] ref string apstrEnrollName) | |
| Functionality | A single string type registration data, equivalent to "USBGetOneEnrollData" | |
| Parameter | anHandleIndex |  |
| apEnrollNumber | Registration number |
| apnBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| apnMachinePrivilege | Variable pointer to the authorization of the registrants |
| apEnrollData | Variable pointer to the enrolled data |
| .apnPassWord | User PassWord |
| apnEnableFlag | User enabled flags |
| apstrEnrollName | User Name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 The directive is similar to "GetEnrollDataWithString" function, the difference is that USBGetOneEnrollDataWithString for usb communication | |

### USBSetOneEnrollData

|  |  |  |
| --- | --- | --- |
| Type | long USBSetOneEnrollData(  long anEnrollNumber,  long anBackupNumber,  long anMachinePrivilege,  long\* apnEnrollData,  long anPassWord,  long anEnableFlag,  BSTR anEnrollName) | |
| Functionality | * Format individual user data in PC memory | |
| Parameter | apEnrollNumber | Registration number |
| anBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| anMachinePrivilege | Variable pointer to the authorization of the registrants |
| apnEnrollData | Variable pointer to the enrolled data |
| anPassWord | Variable pointer of data relating to password or cards |
| anEnableFlag | User enabled flags |
| anEnrollName | User Name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1.This instruction is similar to the PutEnrollData function except that it is only used for usb drive communication  2. Please refer to "EnableUser" for the explanation of "anEnableFlag". | |

### USBSetOneEnrollData\_StringID

|  |  |  |
| --- | --- | --- |
| Type | public static extern int FK\_USBSetOneEnrollData\_StringID(int anHandleIndex, string apEnrollNumber, int anBackupNumber, int anMachinePrivilege, byte[] apEnrollData, int anPassWord, int anEnableFlag, string astrEnrollName); | |
| Functionality | Format single user data in PC memory (support string number) | |
| Parameter | anHandleIndex |  |
| anEnrollNumber | Registration number |
| anBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| anMachinePrivilege | Variable pointer to the authorization of the registrants |
| apEnrollData | Variable pointer to the enrolled data |
| anEnableFlag | User enabled flags |
| astrEnrollName | User Name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | This instruction is similar to the PutEnrollDataWithString function except that it is only used for usb drive communication. | |

### USBWriteAllEnrollDataToFile

|  |  |  |
| --- | --- | --- |
| Type | long USBWriteAllEnrollDataToFile(BSTR astrFilePath) | |
| Functionality | To file the enrollment data formed in the internal memory of the PC by “USBSetOneEnrollData” or “USBSetOneEnrollDataWithString” | |
| Parameter | astrFilePath | File name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “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. | |

### ReadAllUserID

|  |  |  |
| --- | --- | --- |
| Type | long ReadAllUserID() | |
| 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 “Error Code Table”. | |
| Others | 1 The read information can be got with a command “GetAllUserID”. As for “GetAllUserID”, please refer to “0GetAllUserID”.  2 The command fails to be executed if the enrolled registrant does not exist. | |

### 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 “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 “0 ModifyPrivilege”.  4 For the meaning of “apnEnableFlag”, please refer to “0 EnableUser”. | |

### GetAllUserID\_StringID

|  |  |  |
| --- | --- | --- |
| Type | public static extern int FK\_GetAllUserID\_StringID(  int anHandleIndex,  [MarshalAs(UnmanagedType.LPStr)] ref string apEnrollNumber,  ref int apnBackupNumber,  ref int apnMachinePrivilege,  ref int apnEnableFlag); | |
| Functionality | To get registrants ID information with sting. | |
| 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 “Error Code Table”. | |
| Others | 1 Before the execution of this command, it is necessary to backup the registered data. | |

### EmptyEnrollData

|  |  |  |
| --- | --- | --- |
| Type | long EmptyEnrollData() | |
| 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 “Error Code Table”. | |
| Others | 1 Before the execution of this command, it is necessary to backup the registered data. | |

### ClearKeeperData

|  |  |  |
| --- | --- | --- |
| Type | long ClearKeeperData() | |
| Functionality | Clear all data on equipment including registration data, attendance record, management record | |
| Parameter |  |  |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | Please back up all information before executing this command | |

### BenumbAllManager

|  |  |  |
| --- | --- | --- |
| Type | long BenumbAllManager() | |
| 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 “Error Code Table”. | |
| Others |  |  |

### \*GetVerifyMode

|  |  |  |
| --- | --- | --- |
| Type | long GetVerifyMode(long anEnrollNumber, long \*apnVerifyMode) | |
| Functionality | Gets the authentication method for a single user | |
| Parameter | anEnrollNumber | Registration number |
| apnVerifyMode | Variable pointer of the verification mode |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others |  | |

### \*SetVerifyMode

|  |  |  |
| --- | --- | --- |
| Type | long SetVerifyMode(long anEnrollNumber, long anVerifyMode) | |
| Functionality | Set authentication for individual users | |
| Parameter | anEnrollNumber | Registration number |
| apVerifyMode | Variable pointer of the verification mode |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others |  | |

### \*USBGetOneEnrollData\_1

|  |  |  |
| --- | --- | --- |
| Type | long USBGetOneEnrollData\_1(  long\* apnEnrollNumber,  long\* apnBackupNumber,  long\* apnVerifyMode,  long\* apnMachinePrivilege,  long\* apnEnrollData,  long\* apnPassWord,  long\* apnEnableFlag,  BSTR\* apstrEnrollName) | |
| Functionality | Use instructions "USBReadAllEnrollDataFromFile" read user information | |
| Parameter | apnEnrollNumber | Registration number |
| apnBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| apnVerifyMode | Variable pointer of the verification mode |
| apnMachinePrivilege | Variable pointer of the operational authorization of the registrant |
| apnEnrollData | Variable pointer of the fingerprint data |
| apnPassWord | Variable pointer of data relating to password or cards |
| apnEnableFlag | Variable pointer of the flag enabling the registrant to use the device |
| apstrEnrollName | User Name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 the similar and different part of this instruction and GetEnrollData instruction is not directly connected to the device, but only used USB memory. For the instruction of GetEnrollData instruction, please refer to "2.2.1 GettEnrollData".  2 when all the data is retrieved, return the code "RUNERR\_LOG\_END"  3 when the computer without using the command "USBReadAllEnrollDataFromFile" import registration information, order will be failure  4 the meaning of the command "apnEnableFlag" please refer to "2.4.1 EnableUser" | |

### \*USBGetOneEnrollDataWithString\_1

|  |  |  |
| --- | --- | --- |
| Type | long USBGetOneEnrollDataWithString\_1(  long\* apnEnrollNumber,  long\* apnBackupNumber,  long\* apnVerifyMode,  long\* apnMachinePrivilege,  BSTR\* apstrEnrollData,  long\* apnEnableFlag,  BSTR\* apnEnrollName | |
| Functionality | Similar to "USBGetOneEnrollData" instruction, in the form of text column to obtain the registration information | |
| Parameter | apnEnrollNumber | Registration number |
| apnBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| apnVerifyMode | Variable pointer of the verification mode |
| apnMachinePrivilege | Variable pointer of the operational authorization of the registrant |
| apnEnrollData | Variable pointer to the enrolled data |
| apnEnableFlag | Variable pointer of the flag enabling the registrant to use the device |
| apstrEnrollName | User Name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 The only difference between this instruction and "GetEnrollDataWithString" instruction is that it is not connected directly, but only USES USB memory. For the instruction of "GetEnrollDataWithString", please refer to "2.2.2 GetEnrollDataWithString".  2 descr refer to "2.2.9 USBGetOneEnrollData" | |

### \*USBSetOneEnrollData\_1

|  |  |  |
| --- | --- | --- |
| Type | long USBSetOneEnrollData(  long anEnrollNumber,  long anBackupNumber,  long anVerifyMode,  long anMachinePrivilege,  long\* apnEnrollData,  long anPassWord,  long anEnableFlag,  BSTR astrEnrollName) | |
| Functionality | In order to convert the user's operation permission and registration data into USB memory files, formalize on PC memory | |
| Parameter | anEnrollNumber | Registration number |
| anBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| anVerifyMode | Variable pointer of the verification mode |
| apMachinePrivilege | Operational authorization of the registrants |
| apnEnrollData | Variable pointer of the fingerprint data |
| anPassWord | Variable pointer of data relating to password or cards |
| apEnableFlag | User enabled flags |
| astrEnrollName | User Name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 This instruction is similar to the PutEnrollData instruction in that it does not directly connect to the device, but only USES USB memory. For the instruction of PutEnrollData instruction, please refer to "2.2.3 PutEnrollData".  2 Please refer to "2.4.1 EnableUser" for the explanation of "anEnableFlag". | |

### \*USBSetOneEnrollDataWithString\_1

|  |  |  |
| --- | --- | --- |
| Type | long USBSetOneEnrollDataWithString\_1(  long anEnrollNumber,  long anBackupNumber,  long anVerifyMode,  long anMachinePrivilege,  BSTR astrEnrollData,  long anEnableFlag,  BSTR astrEnrollName) | |
| Functionality | Like USBSetOneEnrollData instruction, the registration information in the form of text columns | |
| Parameter | anEnrollNumber | Registration number |
| anBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| anVerifyMode | Variable pointer of the verification mode |
| apMachinePrivilege | Operational authorization of the registrants |
| astrEnrollData | Variable pointer to the enrolled data |
| apEnableFlag | User enabled flags |
| astrEnrollName | User Name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 This instruction is similar and different from PutEnrollDataWithString instruction in that it does not directly connect to the device, but only USES USB memory. For the instruction of PutEnrollDataWithString instruction, please refer to "2.2.4 PutEnrollDataWithString".  2 content please refer to the "2.2.9 USBSetOneEnrollData" | |

### \*USBReadAllEnrollDataFromFile\_Color

|  |  |  |
| --- | --- | --- |
| Type | long USBReadAllEnrollDataFromFile\_Color(BSTR astrFilePath) | |
| Functionality | Read the registration data in USB to PC memory for analysis | |
| Parameter | astrFilePath | File Path |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 command cannot be executed when file structure is not correct  2 Please refer to the relevant files for using USB memory on the device | |

### \*USBWriteAllEnrollDataToFile\_Color

|  |  |  |
| --- | --- | --- |
| Type | long USBWriteAllEnrollDataToFile\_Color(BSTR astrFilePath, long anNewsKind) | |
| Functionality | Created by USBSetOneEnrollData or USBSetOneEnrollDataWithString instructions on PC memory form of registration data files | |
| Parameter | astrFilePath | File Path |
| anNewsKind | NewKind = 0x02 : 60th Chinese Chars  NewKind = 0x01 : 24 Chinese Chars |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 In the instructions before, first get useful USBSetOneEnrollData or USBSetOneEnrollDataWithStrIng instructions do formal data well.  2 Please refer to the relevant files for using USB memory on the device | |

### \*USBGetOneEnrollData\_Color

|  |  |  |
| --- | --- | --- |
| Type | long USBGetOneEnrollData\_Color (  long\* apnEnrollNumber,  long\* apnBackupNumber,  long\* apnMachinePrivilege,  long\* apnEnrollData,  long\* apnPassWord,  long\* apnEnableFlag,  BSTR\* apstrEnrollName,  long anNewsKind) | |
| Functionality | Read the registration data obtained through USBReadAllEnrollDataFromFile instructions. | |
| Parameter | apnEnrollNumber | Registration number |
| apnBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| apnMachinePrivilege | Variable pointer of the operational authorization of the registrant |
| apnEnrollData | Variable pointer of the fingerprint data |
| apnPassWord | Variable pointer of data relating to password or cards |
| apnEnableFlag | Variable pointer of the flag enabling the registrant to use the device |
| apstrEnrollName | User Name |
| anNewsKind | Message setting type |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 This instruction is similar to the GetEnrollData instruction. The different part is not directly connected, only using USB memory. About the instruction of GetEnrollData refer to "2.2.1 GetEnrollData".  2 Return RUNERR\_LOG\_END after retrieving the data.  3 If by USBReadAllEnrollDataFromFile instructions on your computer doesn't read registration information number according to the instructions by failure.  4 Please refer to 2.4.1 EnableUser for the explanation of "apnEnableFlag". | |

### \*USBGetOneEnrollDataWithString\_Color

|  |  |  |
| --- | --- | --- |
| Type | long USBGetOneEnrollDataWithString\_Color (  long\* apnEnrollNumber,  long\* apnBackupNumber,  long\* apnMachinePrivilege,  BSTR\* apstrEnrollData,  long\* apnEnableFlag,  BSTR\* apstrEnrollName,  long anNewsKind) | |
| Functionality | Similar to USBGetOneEnrollData instructions, in the form of text column to obtain the registration information. | |
| Parameter | apnEnrollNumber | Registration number |
| apnBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| apnMachinePrivilege | Variable pointer of the operational authorization of the registrant |
| apstrEnrollData | Variable pointer to the enrolled data |
| apnEnableFlag | Variable pointer of the flag enabling the registrant to use the device |
| apstrEnrollName | User Name |
|  | anNewsKind | Message setting type |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”.. | |
| Others | 1 This instruction is similar to the GetEnrollDataWithString instruction, except that it only USES USB memory instead of direct connection. For the GetEnrollDataWithString instruction, please refer to "2.2.2 GetEnrollDataWithString".  2 descr please reference "2.2.9 USBGetOneEnrollData". | |

### \*USBSetOneEnrollData\_Color

|  |  |  |
| --- | --- | --- |
| Type | long USBSetOneEnrollData\_Color (  long anEnrollNumber,  long anBackupNumber,  long anMachinePrivilege,  long\* apnEnrollData,  long anPassWord,  long anEnableFlag,  BSTR astrEnrollName,  long anNewsKind) | |
| Functionality | For the user's operation rights and registration data into USB memory files, formalized in PC memory. | |
| Parameter | anEnrollNumber | Registration number |
| anBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| anMachinePrivilege | Operational authorization of the registrants |
| apnEnrollData | Variable pointer of the fingerprint data |
| anPassWord | Variable pointer of data relating to password or cards |
| anEnableFlag | User enabled flags |
| astrEnrollName | User Name |
| anNewsKind | Message setting type |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 This instruction and PutEnrollData instruction are similar and different in that they do not connect directly, but only use USB storage. For instructions on PutEnrollData instruction, please refer to 2.2.3 PutEnrollData.  2 Please refer to 2.4.1 EnableUser for the explanation of "anEnableFlag". | |

### \*USBSetOneEnrollDataWithString\_Color

|  |  |  |
| --- | --- | --- |
| Type | long USBSetOneEnrollDataWithString\_Color(  long anEnrollNumber,  long anBackupNumber,  long anMachinePrivilege,  BSTR astrEnrollData,  long anEnableFlag,  BSTR astrEnrollName,  long anNewsKind) | |
| Functionality | Like USBSetOneEnrollData instruction, the registration information in the form of text columns. | |
| Parameter | anEnrollNumber | Registration number |
| anBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| anMachinePrivilege | Operational authorization of the registrants |
| astrEnrollData | Variable pointer to the enrolled data |
| anEnableFlag | User enabled flags |
| astrEnrollName | User Name |
| anNewsKind | Message setting type |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. . | |
| Others | 1 This instruction is similar to PutEnrollDataWithString instruction, the different part is not directly connected, only USB memory. For the instruction of PutEnrollDataWithString, please refer to "2.2.4 PutEnrollDataWithString".  2 please reference "2.2.11 USBSetOneEnrollData". | |

## Management of Recorded Data

### \*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 “Error Code Table”. | |
| Others | 1 The read data can be got by “GetSuperLogData” Please refer to “GetSuperLogData”.  2  anReadMark = 1 permits reading the newly-added recorded data alone.  anReadMark = 0 permits reading all of the recorded data. | |

### \*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 “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. |

### \*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 “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”:   1. : LOG\_ENROLL\_USER - To register general users 2. : LOG\_ENROLL\_MANAGER - To register manager(s) 3. : LOG\_ENROLL\_DELFP - To delete fingerprint data 4. : LOG\_ENROLL\_DELPASS - To delete passwords 5. : LOG\_ENROLL\_DELCARD - To delete card data 6. : LOG\_LOG\_ALLDEL - To delete all the management data 9 : LOG\_SETUP\_SYS - To modify the information about the devices 7. : LOG\_SETUP\_TIME - To modify the time of the devices 8. : LOG\_SETUP\_LOG - To modify the limit values of the management data 9. : 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 | |

### \*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 “Error Code Table”. | |
| Others | 1 Before the execution of this command, it is necessary to backup the management data. | |

### 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 “Error Code Table”. | |
|  | Others | 1 The read data can be got by “GetGeneralLogData”. Please, refer to “0 GetGeneralLogData”.  2 anReadMark = 1 allows to read newly-added recorded data alone.  anReadMark = 0 allows to read all the recorded data. | |

### 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 “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. |

### 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 “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 5. : LOG\_PASSFP\_VERIFY - Verified as fingerprints added to passwords 6. : 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”.)   1. : LOG\_OPEN\_DOOR - The signal of opening the door is transmitted after the verification. 2. : LOG\_CLOSE\_DOOR - The signal of closing the door is transmitted after the verification 3. : LOG\_OPEN\_HAND - The signal of opening the door with the key is transferred. 4. : LOG\_OPEN\_THREAT - The signal of opening the door by verifying threatened fingerprints is transferred. 5. : LOG\_PROG\_OPEN - The signal of opening the door is transferred from the controlling device. 6. : LOG\_PROG\_CLOSE - The signal of closing the door is transferred from the controlling device. 7. : LOG\_OPEN\_IREGAL - The signal of opening the door is illegally transferred. 8. : LOG\_CLOSE\_IREGAL - The signal of closing the door is illegally transferred. 9. : LOG\_OPEN\_COVER - The cover of the device opened 10. : 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”:   1. : LOG\_IOMODE\_IN - Verified with the mode of coming in 2. : LOG\_IOMODE\_OUT - Verified with the mode of going out 3. : LOG\_IOMODE\_IO - Verified with the general mode |

### EmptyGeneralLogData

|  |  |  |
| --- | --- | --- |
| Type | long EmptyGeneralLogData() | |
| 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 “Error Code Table”. | |
| Others | 1 | It is necessary to backup the data relating to incoming and outgoing before the execution of this command. |

### \* GetGeneralLogData\_1

|  |  |  |
| --- | --- | --- |
| Type | long GetGeneralLogData\_1(  long\* apnEnrollNumber,  long\* apnVerifyMode,  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 “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”.)   1. : LOG\_OPEN\_DOOR - The signal of opening the door is transmitted after the verification. 2. : LOG\_CLOSE\_DOOR - The signal of closing the door is transmitted after the verification 3. : LOG\_OPEN\_HAND - The signal of opening the door with the key is transferred. 4. : LOG\_OPEN\_THREAT - The signal of opening the door by verifying threatened fingerprints is transferred. 5. : LOG\_PROG\_OPEN - The signal of opening the door is transferred from the controlling device. 6. : LOG\_PROG\_CLOSE - The signal of closing the door is transferred from the controlling device. 7. : LOG\_OPEN\_IREGAL - The signal of opening the door is illegally transferred. 8. : LOG\_CLOSE\_IREGAL - The signal of closing the door is illegally transferred. 9. : LOG\_OPEN\_COVER - The cover of the device opened   : 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”:   1. : LOG\_IOMODE\_IN - Verified with the mode of coming in 2. : LOG\_IOMODE\_OUT - Verified with the mode of going out 3. : LOG\_IOMODE\_IO - Verified with the general mode   5 \* If the user\_id of the uploaded record is 0, it means that the person (the illegal user) who has not been fingerprinted punched in the machine | |

### \* 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 “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”:   1. : LOG\_ENROLL\_USER - To register general users 2. : LOG\_ENROLL\_MANAGER - To register manager(s) 3. : LOG\_ENROLL\_DELFP - To delete fingerprint data 4. : LOG\_ENROLL\_DELPASS - To delete passwords 5. : LOG\_ENROLL\_DELCARD - To delete card data 6. : LOG\_LOG\_ALLDEL - To delete all the management data 9 : LOG\_SETUP\_SYS - To modify the information about the devices 7. : LOG\_SETUP\_TIME - To modify the time of the devices 8. : LOG\_SETUP\_LOG - To modify the limit values of the management data 9. : 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 | |

### \*GetRealTimeInfo

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

### \*SetRealTimeInfo

|  |  |  |
| --- | --- | --- |
| Type | long SetRealTimeInfo(long\* apRealTimeInfo) | |
| Functionality | To write into machines the waiting time for transfer of blocks and sectors of time for automatic uploading of transactions | |
| Parameter | apRealTimeInfo | Setting data |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |

### LoadGeneralLogDataByDate

|  |  |  |
| --- | --- | --- |
| Type | long FK\_LoadGeneralLogDataByDate(  DateTime anStartDateTime,  DateTime anEndDateTime); | |
| Functionality | Get Logs by time period | |
| Parameter | anStartDateTime | Start Time |
| anEndDateTime | End Time |
| Type | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |

## Management of Devices

### \*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 “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. | |

### \*PowerOnAllDevice

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

### \*PowerOffDevice

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long PowerOffDevice() | | |
| 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 “Error Code Table”. | | |
| Others | 1 | After the execution of this command, the device is disconnected and powered off. | |

### 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 “Error Code Table”. | | |
| Others | 1 |  | |

### 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 “Error Code Table”. | | |
| Others | 1 |  | |

### 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 “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. 6. : GET\_ASLOGS - The number of the entire management existing –data. 7. : GET\_AGLOGS - The number of the entire Income/Outgoing existing-data. 9 : GET\_CARDS - The number of card data existing currently | |

### 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 “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 | |

### 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 “Error Code Table”. | | |
| Others | 1 | The values of “anInfoIndex” are the same as “0 GetDeviceInfo” gives. | |

### GetProductData

|  |  |  |
| --- | --- | --- |
| Type | long GetProductData(long anProductIndex, BSTR\* 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 “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 | |

### \*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 “Error Code Table”. | | |
| Others | 1 |  | |

### \*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 “Error Code Table”. | | |
| Others | 1 |  | |

### \*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 “Error Code Table”. | | |
| Others | 1 |  | |

## Management of Registrants` Information

### \*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 “Error Code Table”. | |
| Others | 1 anEnableFlag = 0 stands for impossibility of the use; anEnableFlag = 1 possibility. | |

### 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 “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”:   1. : MP\_NONE - General user (can only be verified through the device.) 2. : MP\_ALL - Manager (can operate the device.) | |

### GetUserName

|  |  |  |
| --- | --- | --- |
| Type | long GetUserName(long anEnrollNumber, BSTR\* 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 “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. | |

### SetUserName

|  |  |  |
| --- | --- | --- |
| Type | long SetUserName(long anEnrollNumber, BSTR astrUserName) | |
| Functionality | To assign a name to the registrant | |
| Parameter | anEnrollNumber | Registration number |
| astrUserName | Variable pointer containing the name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “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. | |

### \*GetNewsMessage

|  |  |  |
| --- | --- | --- |
| Type | long GetNewsMessage(long anNewsId, BSTR\* 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 “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). | |

### \*SetNewsMessage

|  |  |  |
| --- | --- | --- |
| Type | long SetNewsMessage(long anNewsId, BSTR astrNews) | |
| Functionality | To set a message in the device | |
| Parameter | anNewsId | ID number of the message |
| astrNews | Variable pointer of the message data |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1 For the details, please refer to “0 GetNewsMessage”. | |

### \*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 “Error Code Table”. | |
| Others | 1 “apnNewsId” is a value to be set under “0 SetNewsMessage”. | |

### \*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 “Error Code Table”. | |
| Others | 1 “apnNewsId” is a value to be set under “0 SetNewsMessage”. | |

## Management of Bells

### 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 “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 “0 BELLINFO Structure”. | |

### \*GetBellTimeWithString

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long GetBellTimeWithString(long\* apnBellCount, BSTR\* 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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetBellTime”. | |

### 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 “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 “0 BELLINFO Structure”. | |

### \*SetBellTimeWithString

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type | | long SetBellTimeWithString(long anBellCount, BSTR astrBellInfo) | | | |
|  | Functionality | | Equal to a command “SetBellTime”, it sets the bell-relating information in the form of strings. | | | |
| Parameter | | anBellCount | | | Times of a bell ringing |
| astrBellInfo | | | 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 “Error Code Table”. | | | |
| Others | | 1 | | For the details, please refer to “0 SetBellTime”. | |

## \*Post & Shift Management

### GetOneShiftInfo

|  |  |  |
| --- | --- | --- |
| Type | long GetOneShiftInfo(  long anShiftNumber,  long\* apShiftSHour,  long\* apShiftSMinute,  long\* apShiftEHour,  long\* apShiftEMinute,  BSTR\* apstrShiftName) | |
| Functionality | Function “GetOneShiftInfo” is used to read out the information on shifts set on the terminal. | |
| Parameter | anShiftNumber | Shift Number |
| apShiftSHour | Shift start hour |
| apShiftSMinute | Shift start minute |
| apShiftEHour | Shift end hour |
| apShiftEMinute | Shift end minute |
| apstrShiftName | Shift name |
| . Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |

### SetOneShiftInfo

|  |  |  |
| --- | --- | --- |
| Type | long SetOneShiftInfo(  long anShiftNumber,  long anShiftSHour,  long anShiftSMinute,  long anShiftEHour,  long anShiftEMinute,  BSTR astrShiftName) | |
| Functionality | Function “SetOneShiftInfo” is used to import to the terminal the information on shifts set on the PC. | |
| Parameter | anShiftNumber | Shift Number |
| anShiftSHour | Shift start hour |
| anShiftSMinute | Shift start minute |
| anShiftEHour | Shift end hour |
| anShiftEMinute | Shift end minute |
| astrShiftName | Shift name |
| . Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |

### GetOnePostInfo

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | long GetOnePostInfo(  long anPostNumber,  BSTR\* apStrPostName,  long\* apShiftNumber1,  long\* apShiftNumber2,  long\* apShiftNumber3,  long\* apShiftNumber4) | |
| Functionality | Function “GetOnePostInfo” is used to import from the terminal to the PC the information on departments. | |
| Parameter | anPostNumber | Post number |
| apStrPostName | Post name |
| apShiftNumber1 | Shift1 nimber |
| apShiftNumber2 | Shift2 nimber |
| apShiftNumber3 | Shift3 nimber |
| apShiftNumber4 | Shift4 nimber |
| . Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |

### SetOnePostInfo

|  |  |  |
| --- | --- | --- |
| Type | long SetOnePostInfo(  long anPostNumber,  BSTR astrPostName,  long anShiftNumber1,  long anShiftNumber2,  long anShiftNumber3,  long anShiftNumber4) | |
| Functionality | Function “SetOnePostInfo” is used to import to the terminal the information on departments set on the PC. | |
| Parameter | anPostNumber | Post Number |
| astrPostName | Post Name |
| anShiftNumber1 | Shift1 Number |
| anShiftNumber2 | Shift2 Number |
| anShiftNumber3 | Shift3 Number |
| anShiftNumber4 | Shift4 Number |
| . Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |

### GetUserInfo

|  |  |  |
| --- | --- | --- |
| Type | long GetUserInfo(  long anEnrollNumber,  BSTR\* apstrUserName,  long\* apNewKind,  long\* apVerifyMode,  long\* apPostID,  long\* apShiftNumber1,  long\* apShiftNumber2,  long\* apShiftNumber3,  long\* apShiftNumber4) | |
| Functionality | Function “GetUserInfo” is used to import to the PC the information on users set on the terminal. | |
| Parameter | anEnrollNumber | Registration number |
| apstrUserName | User name |
| apNewKind | News Kind |
| apVerifyMode | Verify Mode |
| apPostID | Post ID |
| apShiftNumber1 | Shift1 Number |
| apShiftNumber2 | Shift2 Number |
| apShiftNumber3 | Shift3 Number |
| apShiftNumber4 | Shift4 Number |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |

### SetUserInfo

|  |  |  |
| --- | --- | --- |
| Type | long SetUserInfo(  long anEnrollNumber,  BSTR astrUserName,  long anNewKind,  long anVerifyMode,  long anPostID,  long anShiftNumber1,  long anShiftNumber2,  long anShiftNumber3,  long anShiftNumber4) | |
| Functionality | Function “SetUserInfo” is used to import to the terminal the information on users set on the PC. | |
| Paramter | anEnrollNumber | Registration number |
| astrUserName | User name |
| anNewKind | News Kind |
| anVerifyMode | Verify Mode |
| anPostID | Post ID |
| anShiftNumber1 | Shift1 Number |
| anShiftNumber2 | Shift2 Number |
| anShiftNumber3 | Shift3 Number |
| anShiftNumber4 | Shift4 Number |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |

### GetPostShiftInfo

|  |  |  |
| --- | --- | --- |
| Type | long GetPostShiftInfo(long\* apPostShiftInfo, long\* apnPostShiftInfoLen) | |
| Functionality | This function gets the department shift information set in the currently connected device with a specific structure. | |
| Parameter | apPostShiftInfo | A pointer to the structure that specifies department and shift information. |
| apnPostShiftInfoLen | A pointer to a variable that receives the size of the department and shift information structure |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. . | |

### SetPostShiftInfo

|  |  |  |
| --- | --- | --- |
| Type | long SetPostShiftInfo(long\* apPostShiftInfo, long anPostShiftInfoLen) | |
| Functionality | Set up department and shift information of attendance machine. | |
| Parameter | apPostShiftInfo | The index of department and shift information structure |
| anPostShiftInfoLen | Size of department and shift structure. |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |

### GetUserInfoEx

|  |  |  |
| --- | --- | --- |
| Type | long GetUserInfoEx(  long anEnrollNumber,  long\* apUserInfo,  long\* apUserInfoLen) | |
| Functionality | Obtain the registered user information of the attendance machine.User information includes the following contents:  The user's name;User's password and ID card number;User's device operation rights;  The user's department number;Number of shifts assigned each day for a month. | |
| Parameter | anEnrollNumber | Registration number |
| apUserInfo | A pointer to a formatted structure that holds user information. |
| apUserInfoLen | A pointer to a variable that receives the size of the user information structure. |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. . | |

### SetUserInfoEx

|  |  |  |
| --- | --- | --- |
| Type | long SetUserInfoEx(  long anEnrollNumber,  long\* apUserInfo,  long anUserInfoLen) | |
| Functionality | Set the attendance machine user information。 | |
| Parameter | anEnrollNumber | Registration number 。 |
| apUserInfo | A pointer to a formatted structure that holds user information. |
| apUserInfoLen | A pointer to a variable that receives the size of the user information structure. |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. . | |

## Photo Management

### GetEnrollPhoto

|  |  |  |
| --- | --- | --- |
| Type | Public static extern int FK\_GetEnrollPhoto(  int nHandleIndex,  UInt32 anEnrollNumber,  byte[] abytPhotoImage,  ref int anPhotoLength); | |
| Functionality | To get user photo by enroll number | |
| Parameter | anEnrollNumber | Enroll Number |
| abytPhotoImage | Photo image of Byte array |
| anPhotoLength | Photo length |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others |  | |

### SetEnrollPhoto

|  |  |  |
| --- | --- | --- |
| Type | Public static extern int FK\_SetEnrollPhoto(  int nHandleIndex,  UInt32 anEnrollNumber,  byte[] abytPhotoImage,  ref int anPhotoLength); | |
| Functionality | To set user photo by enroll number | |
| Parameter | anEnrollNumber | Enroll Number |
| abytPhotoImage | Photo image of Byte array |
| anPhotoLength | Photo length |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others |  | |

### DeletetEnrollPhoto

|  |  |  |
| --- | --- | --- |
| Type | Public static extern int FK\_DeletetEnrollPhoto(  int nHandleIndex,  UInt32 anEnrollNumber); | |
| Functionality | To delete user photo by enroll number | |
| Parameter | anEnrollNumber | Enroll Number |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others |  | |

## Control of Doors

Some of the following functions are not supported in some models.

### 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 “Error Code Table”. | | |
| Others | 1 | The following values are returned to “apnStatusVal”:   1. : DOOR\_CONTROLRESET - control state of door by device. 2. : DOOR\_OPEND - Door opened 3. : DOOR\_CLOSED – Door closed 4. : DOOR\_COMMNAD- by the command for control of doors, door opend for some time and closed. | |

### SetDoorStatus

|  |  |  |  |
| --- | --- | --- | --- |
| Type | public static extern int FK\_HS\_SetTimeZone(int anHandleIndex, byte[] abytOneTimeZone); | | |
| 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 “Error Code Table”. | | |
| Others | 1 | for the meanings of “anStatusVal”, refer to “0 GetDoorStatus”. | |

### GetTimeZone

|  |  |  |
| --- | --- | --- |
| Type | public static extern int FK\_HS\_GetTimeZone(int anHandleIndex, byte[] abytOneTimeZone); | |
| Functionality | To get the information about the time zone for opening and closing the door | |
| Parameter | abytOneTimeZone | One Time Zone with Byte array |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1. The Normal machine support 255 time zones，please refer to actual machine | |

### SetTimeZone

|  |  |  |  |
| --- | --- | --- | --- |
| Type | public static extern int FK\_HS\_SetTimeZone(int anHandleIndex, byte[] abytOneTimeZone); | | |
| Functionality | To set the information about the time zone for opening and closing the door | | |
| Parameter | abytOneTimeZone | | One Time Zone with Byte array |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | | |
| Others |  |  | |

### GetUserWeekPassTime

|  |  |  |
| --- | --- | --- |
| Type | public static extern int FK\_HS\_GetUserWeekPassTime(int anHandleIndex, byte[] abytUserWeekPassTime); | |
| Functionality | To get the information about the user pass time zone for a week | |
| Parameter | abytUserWeekPassTime | Byte Week Pass Time for User |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1.When time zone is 0，it indicate the user do not open the door | |

### SetUserWeekPassTime

|  |  |  |  |
| --- | --- | --- | --- |
| Type | public static extern int FK\_HS\_GetUserWeekPassTime(int anHandleIndex, byte[] abytUserWeekPassTime); | | |
| Functionality | To set the information about the user pass time zone for a week | | |
| Parameter | abytUserWeekPassTime | | Byte Week Pass Time for User |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | | |
| Others |  |  | |

### FK\_ExtCommand

|  |  |  |
| --- | --- | --- |
| Type | public static extern int FK\_ExtCommand(  int anHandleIndex,  byte[] abytCmdStruct) | |
| Functionality | Extension Cmd by byte Structure cmd | |
| Parameter | abytCmdStruct | Byte Structure cmd |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. | |
| Others | 1.It’s used to read/write door info by enroll id,please refer to sample | |

### \*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 “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 “PASSCTRLTIME **Structure**”.  3 As the length of “apnPassTime”, “anPassTimeSize” helps API decide that the structure is long enough. | |

### \*GetPassTimeWithString

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long GetPassTimeWithString(long anPassTimeID, BSTR\* 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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetPassTime”. | |

### \*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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetPassTime”. | |

### \*SetPassTimeWithString

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long SetPassTimeWithString(long anPassTimeID, BSTR astrPassTime) | | |
| 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 |
| astrPassTime | | 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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetPassTime”. | |

### \*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 “Error Code Table”. | |
| Others | 1 For the meaning of “apnGroupID”, please refer to “0 GetGroupPassTime”.  2 “apnPassTimeID” is a array-typed batch structure of ID numbers assigned to the registrants. For its definition, please refer to “0 USERPASSINFO Structure”; for the meanings of the ID numbers, refer to “0 GetPassTime”.  3 As the length of “apnPassTime”, “anPassTimeSize” helps API determine whether the structure is long enough. | |

### \*GetUserPassTimeWithString

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long GetUserPassTimeWithString(  long anEnrollNumber,  long\* apnGroupID,  BSTR\* 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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetUserPassTime”. | |

### \*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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetUserPassTime”. | |

### \*SetUserPassTimeWithString

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long SetUserPassTimeWithString(  long anEnrollNumber,  long anGroupID,  BSTR astrPassTimeID) | | |
| Functionality | Equal to command “SetUserPassTime”, it contains the ID number structure in the form of strings. | | |
| Parameter | anEnrollNumber | | Registration number |
| anGroupID | | Group number |
| astrPassTimeID | | 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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetUserPassTime”. | |

### \*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 “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 “0 GROUPPASSINFO Structure”; for the meanings of ID numbers, refer to “0 GetPassTime”. | |
| 3 | As the length of “apnPassTimeID”, “anPassTimeIDSize” helps API determine whether the structure is long enough. | |

### \*GetGroupPassTimeWithString

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long GetGroupPassTimeWithString(  long anGroupID,  BSTR\* 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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetGroupPassTime”. | |

### \*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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetGroupPassTime”. | |

### \*SetGroupPassTimeWithString

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long SetGroupPassTimeWithString(    long anGroupID,  BSTR astrPassTimeID) | | |
| Functionality | Equal to command “SetGroupPassTime”, it contains ID number structures in the form of strings. | | |
| Parameter | anGroupID | | Group number |
| astrPassTimeID | | 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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetGroupPassTime”. | |

### \*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 “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 “0 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. | |

### \*GetGroupMatchWithString

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long GetGroupMatchWithString(BSTR\* 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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetGroupMatch”. | |

### \*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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetGroupMatch”. | |

### \*SetGroupMatchWithString

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long SetGroupMatchWithString(BSTR astrGroupMatch) | | |
| Functionality | Equal to command “SetGroupMatch”, it contains the union structure in the form of strings. | | |
| Parameter | astrGroupMatch | | 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 “Error Code Table”. | | |
| Others | 1 | For the details, please refer to “0 GetGroupMatch”. | |

## \*Summer time management (deleted)

This feature is not available on all versions.

### GetAdjustInfo

|  |  |
| --- | --- |
| Type | long GetAdjustInfo(  long\* apnAdjustedState,  long\* apnAdjustedMonth,  long\* apnAdjustedDay,  long\* apnAdjustedHour,  long\* apnAdjustedMinute,  long\* apnRestoredState,  long\* apnRestoredMonth,  long\* apnRestoredDay,  long\* apnRestoredHour,  long\* apnRestoredMinute) |
| Functionality | Gets the summer time status. |
| Parameter | A parameter indicating a status value |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. . |
| Others | "ApnAdjustedState, apnRestoredState" : None: no change.  -1: an hour earlier.  +1: one hour late. |

### SetAdjustInfo

|  |  |
| --- | --- |
| Type | long SetAdjustInfo(  long anAdjustedState,  long anAdjustedMonth,  long anAdjustedDay,  long anAdjustedHour,  long anAdjustedMinute,  long anRestoredState,  long anRestoredMonth,  long anRestoredDay,  long anRestoredHour,  long anRestoredMinute) |
| Functionality | Operate the summer time system. |
| Parameter | Status value (refer to 3.1.6 structure) |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. . |
| Others | "ApnAdjustedState, apnRestoredState" : None: no change.  -1: an hour earlier.  +1: one hour late. |

## \* Equipment information management

### GetServerNetInfo

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long GetServerInfo(  BSTR\* apstrServerIPAddress,  long\* apnServerPort,  long\* apnServerRequest) | | |
| Functionality | Read the information from the device server | | |
| Parameter | apstrServerIPAddress | | Server IP Address |
| apnServerPort | | Server Port |
| apnServerRequest | | Set time |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. . | | |
| Others | 1 |  | |

### SetServerNetInfo

|  |  |  |  |
| --- | --- | --- | --- |
| Type | long SetServerNetInfo(  BSTR astrServerIPAddress,  long anServerPort,  long anServerRequest) | | |
| Functionality | Write the server information to the device | | |
| Parameter | astrServerIPAddress | | Server IP Address |
| anServerPort | | Server Port |
| anServerRequest | | Set time |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “Error Code Table”. . | | |
| Others | 1 |  | |

### SetUSBModel

|  |  |  |
| --- | --- | --- |
| Type | void SetUSBModel(long anModel) | |
| Functionality | Specify which type of device to write to (or read from) a USB Flash Flash drive | |
| Parameter | anModel | The parameter anModel is a long variable that represents model information |

## Card/Password(Refer to CardPwdReference.cs)

### string card to byte array

|  |  |  |
| --- | --- | --- |
| Type | public static byte[] getCard(string card) | |
| Functionality | To convert string card to byte array, | |
| Paras | card | Card NO |
| Return | Byte array | |

### string password to byte array

|  |  |  |
| --- | --- | --- |
| Type | public static byte[] getPWD(string pwd) | |
| Functionality | To convert password card to byte array | |
| Paras | card | String NO |
| Return | Byte array | |

Description: according to the class library CardPwdReference.cs, the card/password of string type is converted into byte array, byte data is put into byte array of corresponding length, and finally the data is sent to the device through FK\_PutEnrollData function.

eg:

if (vBackupNumber == 10)

{

byte[] CurEnrollData = CardPwdReference.getPWD(Password.Text);

Array.Clear(mbytCurEnrollData, 0, mbytCurEnrollData.Length);

Array.Copy(CurEnrollData, mbytCurEnrollData, PWD\_DATA\_SIZE);

}

vnResultCode = FKAttendDLL.FK\_PutEnrollData(

FKAttendDLL.nCommHandleIndex,

vEnrollNumber,

vBackupNumber,

vPrivilege,

mbytCurEnrollData,

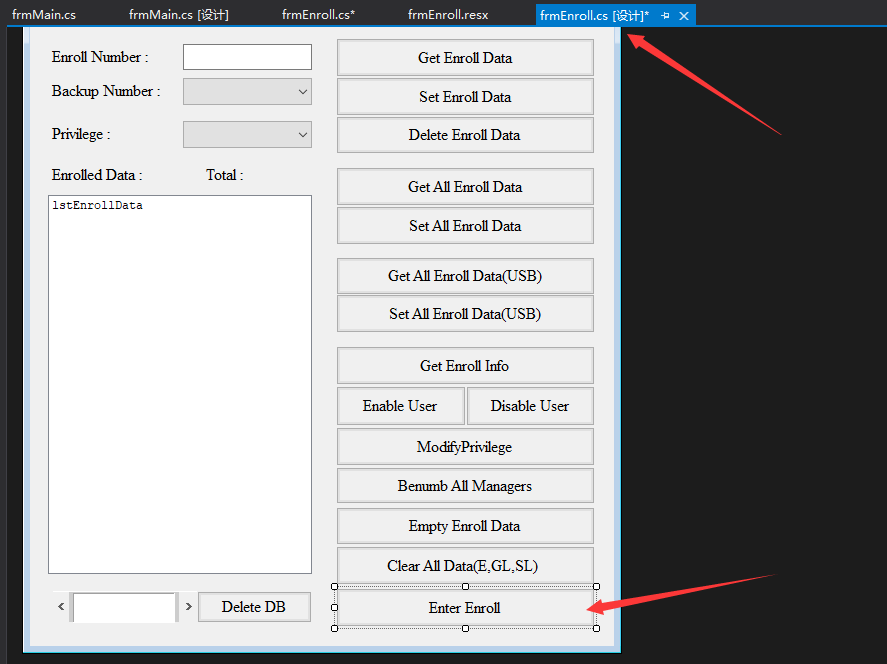
mnCurPassword);

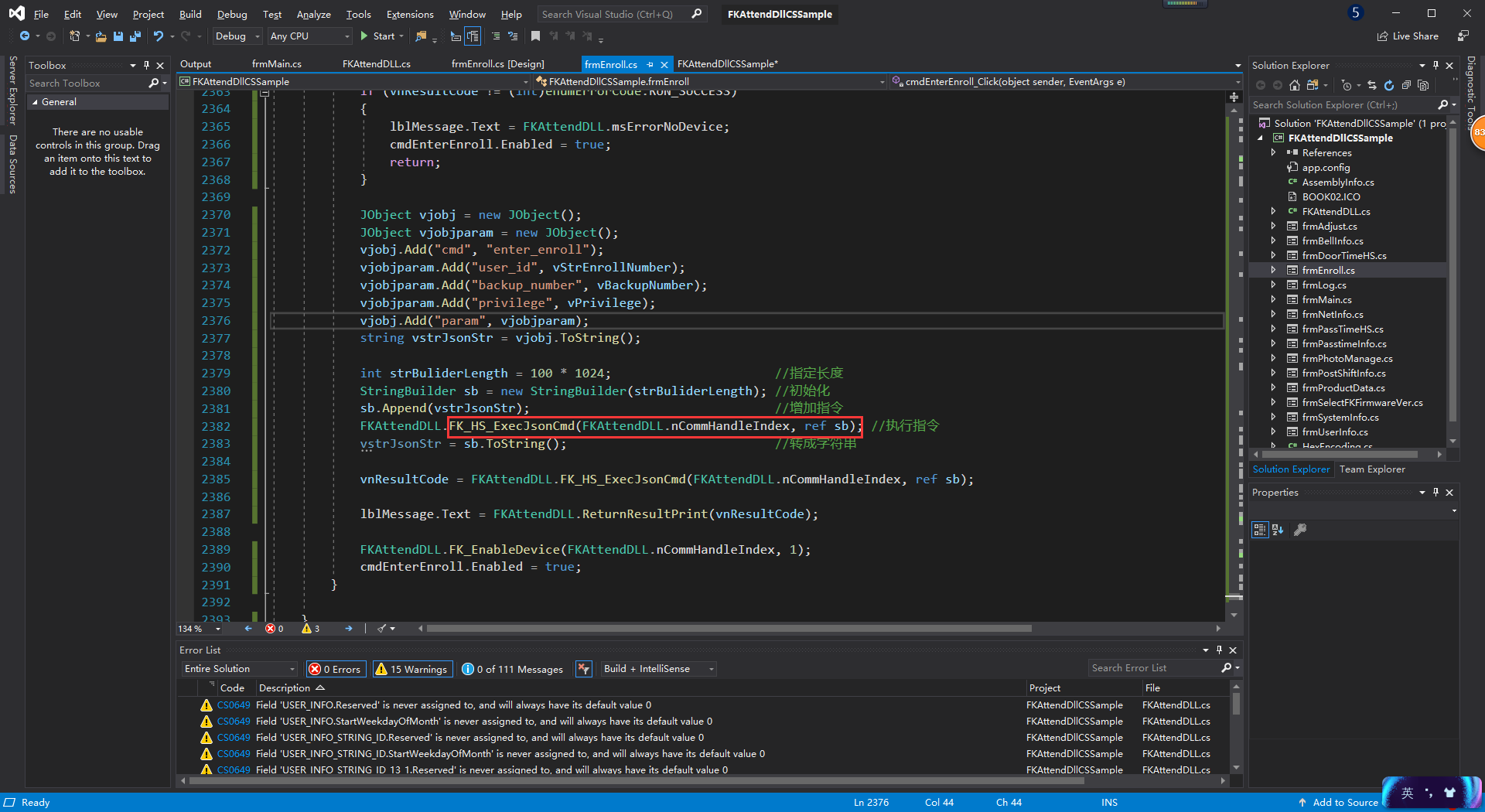
### Show card and password(refer to function of below)

|  |  |  |
| --- | --- | --- |
| Type | public static string GetCardOrPwdWithStr(byte[] byteCurrEnrollDate) | |
| Functionality | To convert byte array to string card /password. | |
| Paras | byteCurrEnrollDate | Data byte array of Enroll |
| Return | String card/password | |

## The Common Interface of JsonCommand

Please refer to frmEnroll.cs in the sample,the interface **FK\_HS\_ExecJsonCmd** at “Enter Enroll”





### Import function

[DllImport("FK623Attend.dll", CharSet = CharSet.Ansi)]

public static extern int FK\_HS\_ExecJsonCmd(

int anHandleIndex,

[MarshalAs(UnmanagedType.LPStr)]ref StringBuilder apJsonStr);

//return **1** mean the cmd excecute successfully

### Enter Enroll

Go to an employee registration page

Coding：

JObject vjobj = new JObject();

JObject vjobjparam = new JObject();

vjobj.Add("cmd", "enter\_enroll");//**enter\_enroll**为指令类型

vjobjparam.Add("user\_id", vStrEnrollNumber);

vjobjparam.Add("backup\_number", vBackupNumber);

vjobjparam.Add("privilege", vPrivilege);

vjobj.Add("param", vjobjparam);

string vstrJsonStr = vjobj.ToString();

int strBuliderLength = 100 \* 1024; // Specify the length

StringBuilder sb = new StringBuilder(strBuliderLength); //Init

sb.Append(vstrJsonStr); // Add Cmd

vnResultCode = FKAttendDLL.FK\_HS\_ExecJsonCmd(FKAttendDLL.nCommHandleIndex, ref sb); // Execution Instruction

### Set Lock Control

JObject vjobj = new JObject();

JObject vjobjparam = new JObject();

vjobj.Add("cmd", "**lock\_control\_set**");

vjobjparam.Add("Receive\_Interval", "5");// Receive\_Interval 1-255

vjobjparam.Add("Allow\_LateTime", "0");// Allow\_LateTime 0-60

vjobjparam.Add("Allow\_EarlyTime", "0");// Allow\_EarlyTime 0-60

vjobjparam.Add("Managers", "0");// All Managers 0-10

vjobjparam.Add("Volume", "0");// Volume 0-10

vjobjparam.Add("Glog\_Warning", "0");// Glog\_Warning 0-1000

vjobjparam.Add("Show\_ResultTime", "1");// Show\_ResultTime 0-30

vjobjparam.Add("Reverify\_Time", "0");// Reverify\_Time 0-60

vjobjparam.Add("Screensavers\_Time", "0");// Screensavers\_Time 0-60\*24

vjobjparam.Add("Sleep\_Time", "0");/ Sleep\_Time 0-60

vjobjparam.Add("Wiegand\_Type", "34");//Wiegand Type 26,34

vjobjparam.Add("Wiegand\_Output", "0");//Wiegand Output 0,1

vjobjparam.Add("Wiegand\_Input", "0");//Wiegand Input 0,1

vjobjparam.Add("OpenDoor\_Delay", "0");// OpenDoor\_Delay 0-200

vjobjparam.Add("MutiUser", "1");// Muti confirmation User 1-10

vjobjparam.Add("DoorMagnetic\_Type", "no");// DoorMagnetic\_Type "no","close","open"

vjobjparam.Add("DoorMagnetic\_Delay", "10");// DoorMagnetic\_Delay 0-200

vjobjparam.Add("Alarm\_Delay", "1");// Alarm\_Delay 0-255

vjobjparam.Add("Anti-back", "no");// Prevent come back "no","yes"

vjobjparam.Add("Use\_Alarm", "yes");// Tamper alarm "no","yes"

vjobj.Add("param", vjobjparam);

string vstrJsonStr = vjobj.ToString();

int strBuliderLength = 100 \* 1024; //Specify the length

StringBuilder sb = new StringBuilder(strBuliderLength); //Init

sb.Append(vstrJsonStr); //Add Cmd

vnResultCode = FKAttendDLL.FK\_HS\_ExecJsonCmd(FKAttendDLL.nCommHandleIndex, ref sb); // Execution Instruction

### Set Attend Status

JObject vjobj = new JObject();

JObject vjobjparam = new JObject();

vjobj.Add("cmd", "att\_status\_set");

for (int i = 1; i <= 10; i++)

{

vjobjparam.Add("name" + i.ToString(), "上班"); //仅作参考

vjobjparam.Add("enable" + i.ToString(), "no");//"no","yes"

vjobjparam.Add("hour" + i.ToString(), "0");

vjobjparam.Add("minute" + i.ToString(), "0");

}

vjobj.Add("param", vjobjparam);

string vstrJsonStr = vjobj.ToString();

int strBuliderLength = 100 \* 1024; //Specify the length

StringBuilder sb = new StringBuilder(strBuliderLength); //Init

sb.Append(vstrJsonStr); //Add Cmd

vnResultCode = FKAttendDLL.FK\_HS\_ExecJsonCmd(FKAttendDLL.nCommHandleIndex, ref sb); // Execution Instruction

### Get Lock Control Params

JObject vjobj = new JObject();

vjobj.Add("cmd", "lock\_control\_get");

string vstrJsonStr = vjobj.ToString();

int strBuliderLength = 100 \* 1024; //Specify the length

StringBuilder sb = new StringBuilder(strBuliderLength); //Init

sb.Append(vstrJsonStr); //Add Cmd

FKAttendDLL.FK\_HS\_ExecJsonCmd(ConnectDevice.nCommHandleIndex, ref sb); // Execution Instruction

vstrJsonStr = sb.ToString(); //Convert to string

### Get Attend Status

JObject vjobj = new JObject();

vjobj.Add("cmd", "att\_status\_get");

string vstrJsonStr = vjobj.ToString();

FKAttendDLL.FK\_HS\_ExecJsonCmd (ref vstrJsonStr);

int strBuliderLength = 100 \* 1024; // Specify the length

StringBuilder sb = new StringBuilder(strBuliderLength); //Init

sb.Append(vstrJsonStr); //Add Cmd

FKAttendDLL.FK\_HS\_ExecJsonCmd(ConnectDevice.nCommHandleIndex, ref sb); // Execution Instruction

vstrJsonStr = sb.ToString(); //Convert to string

### JsonCommand using attentions

1. When obtaining the commonly used access control parameters and attendance status, there may be some scrambled codes after}, which should be dealt with

2. Use ANSI encoding

## Notes on use of DLL interface

First, the return value of connection functions (e,g FK\_ConnectComm, FK\_ConnectNet) is the number to identify individual connection to the FK machine.

You must pass this value as first argument of most interface functions called after connected.

Second, when you pass string argument to interface function you must pass string value which is ended with zero. This time the type of parameter used to pass string value is defined as 'char\*' and this parameter is the pointer to the string buffer.

When you receive string value as the output parameter of the function, you must allocate enough buffer to receive output string and pass the address of the address of the allocated buffer to the function.

This time the type of parameter used to pass address of the address is defined as 'char\*\*'.

The value passed to function is the pointer to the pointer to the bufffer to receive string.

The code value of string is based on the ANSI code according to the language setting of current Windows OS.

For example :

The interface functions to get user name saved in FK machine are defined individually in OCX and DLL.

|  |
| --- |
| interface function of DLL |
| long FK\_GetUserName(long anHandleIndex, long anEnrollNumber, char\*\* apstrUserName); |

An important note to remember is that you must allocate enough buffer in advance to receive output name string value.

How to call ‘FK\_GetUserName’function declared in DLL using VB6.0

|  |
| --- |
| ‘ -- declaration ----------------------------------------------------  Public Declare Function FK\_GetUserName Lib "FK623Attend" (    ByVal nHandleIndex As Long,    ByVal nEnrollNumber As Long,    ByRef pstrUserName As String) As Long  ‘ ------------------------------------------------------------------    ‘-- example code --------------------------------------------------------  Dim nEnrollNumber As Long  Dim nResultCode As Long  Dim strName As String    strName = Space(256) ‘ allocate 256 byte buffer and initialize as space character.        ‘ The maximum length of name string does not exceed 256 bytes.  nEnrollNumber = Val(Trim(txtEnrollNumber.Text)) nResultCode = FK\_GetUserName( \_  fnCommHandleIndex, \_  nEnrollNumber, \_  strName) |
| How to call ‘FK\_GetUserName’function declared in DLL using VC6.0 |
| //-- Declare Func ---------------------------------------------------- long FP\_EXPORT FK\_GetUserName(long anHandleIndex, long anEnrollNumber, **char\*\* apstrUserName**);  // ------------------------------------------------------------------    //-- Example Coding ------------------------------------------------------- char\* pszTemp = new char[256];  nErrorCode = FK\_GetUserName(m\_nCommHandleIndex, nEnrollNumber, &pszTemp); AfxMessageBox(pszTemp);  delete [] pszTemp; |

# RealSvrOcxTcp.ocx Component

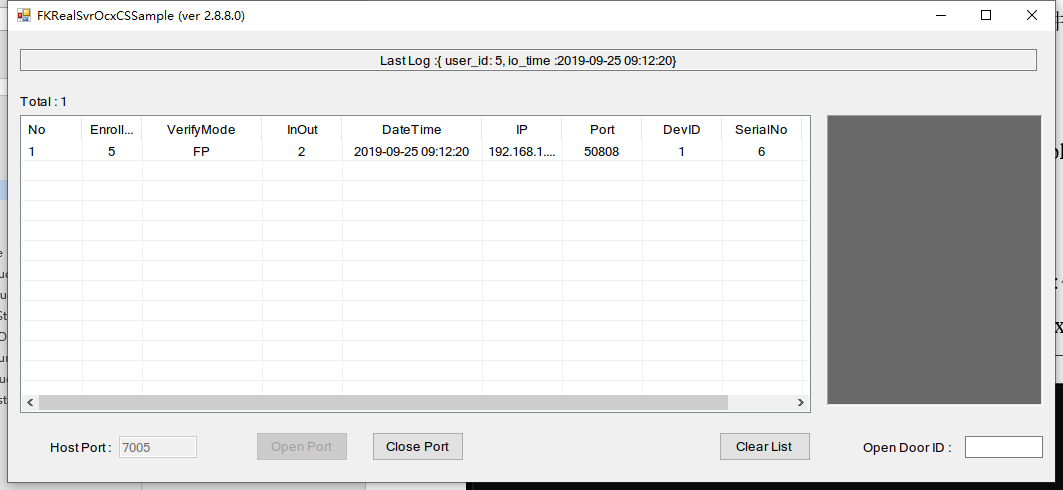
RealSvrOcxTcp.ocx is a Component that is available in VS and is used to accept real-time recording and access alarm information transmitted by the machine, using as follows:

## Introduction to the use of Demo:

1.Run FKRealSvrOcxTcpCSSample.exe in folder Execute

2.Specify port and open it

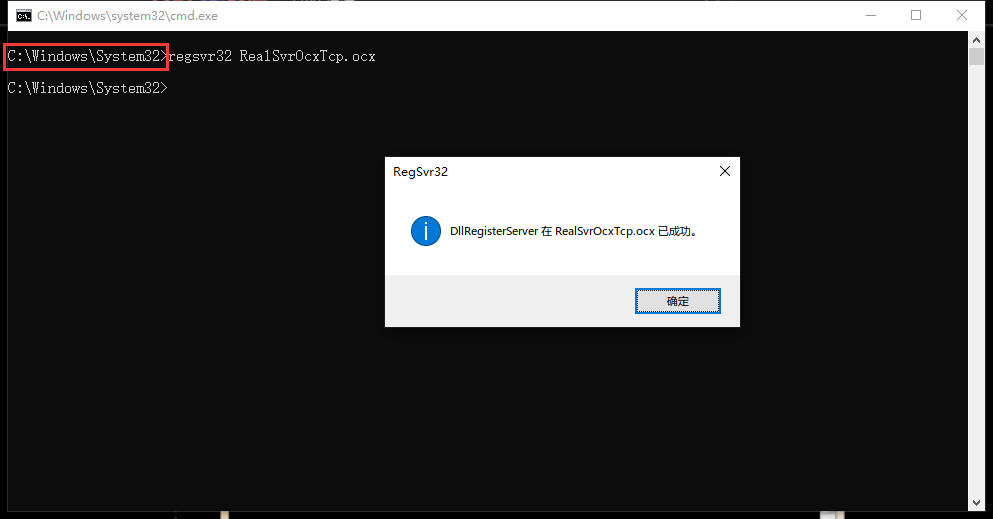
3.After identifying fingerprint in machine ，the form will show a record



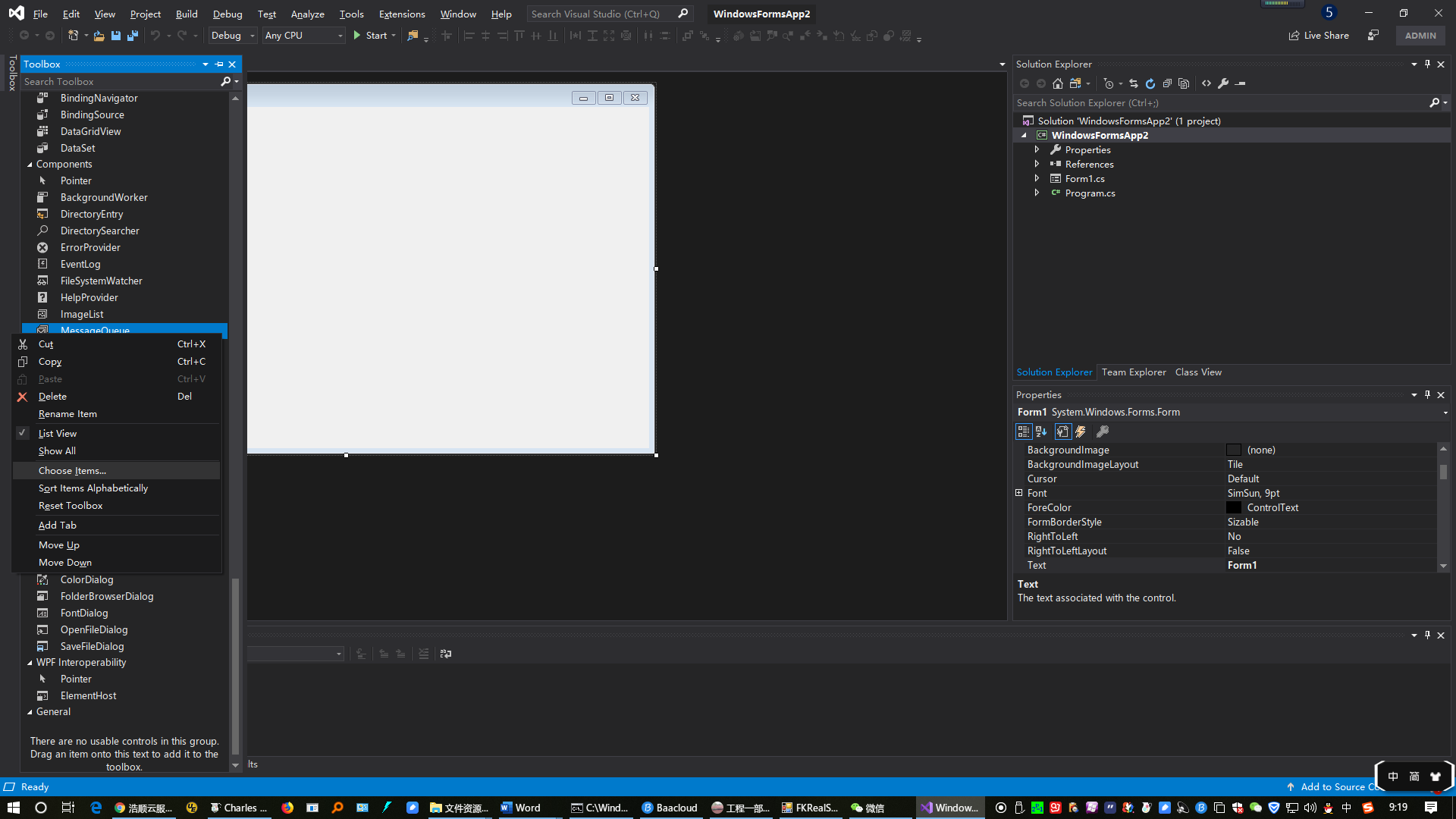
## Introduction to control usage

1. Copy RealSvrOcxTcp.ocx，in floder Execute， to C:\Windows\System32

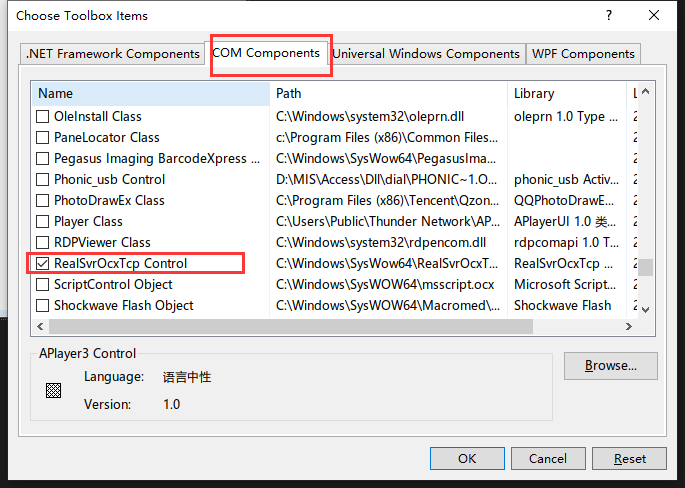
2. Open CMD ，input “ regsvr32 RealSvrOcxTcp.ocx ”，and show register successful



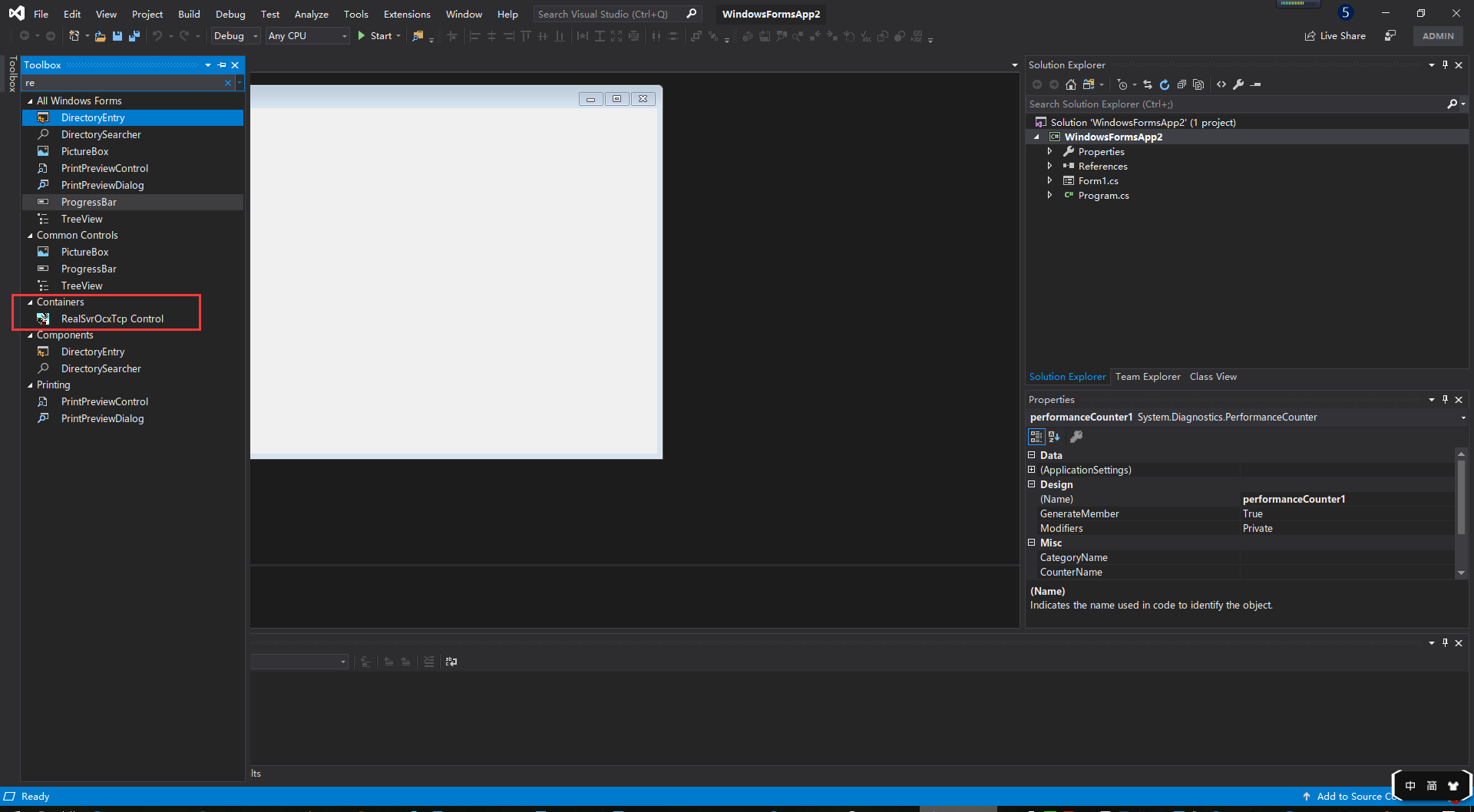
3. In visual studio tool bar，right-click and choose the Choose Items



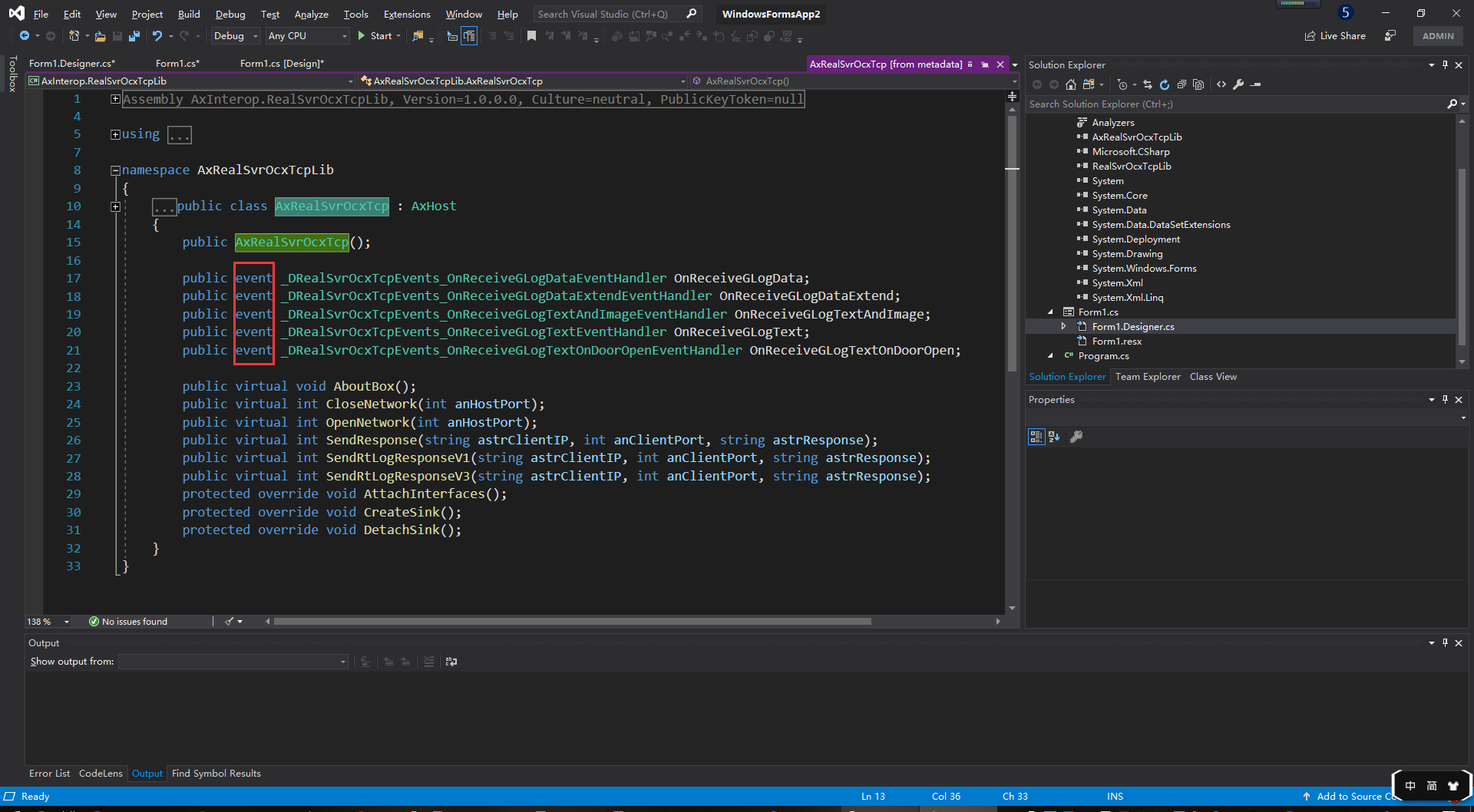
4. In Coms Components ，choose component of Realsvrocxtcp Control



5. Go back to the toolbar to find the control RealSvrOcxTcp Control



6. The control is primarily an event that triggers when a record is received when monitoring is turned on. Samples Code in



# Appendix

## Structures

### 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;

### 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

### 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

### 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

### 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

### 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;

### REALTIMEINFO Structure

#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

### SetUSBModel Constants

#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.

## 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 |

# Version

|  |  |  |
| --- | --- | --- |
| Update time | Dynamic Link Library | Update content |
| 2019-10-10 | FKAttend.dll(2.8.12.227) | Change the content of the InOut field in the record, and the dynamic library FKAttend.dll is updated.Samples has also changed accordingly, refer to the function GetIoModeAndDoorMode. repair string ID length display error. |
| 2019-09-15 | ---Manual update | If the user\_id of the uploaded real-time record is 0, it means that the person who has not been fingerprinted is punching in on the machine |
| 2019-05-25 | FK623Attend.dll(2.8.12.222) | Increase the interface Fk\_loadgenerallogdatabase date retrieves records by time period |
| 2019-03-08 | FK623Attend.dll(2.8.12.221) | Increase the general interface JsonCommand instructions, machine Settings, attendance status change |
| 2018-12-27 | FKAttend.dll(2.8.12.224) | Update GetDeviceStatus, add palm print |
| 2018-12-18 | --- | Description of card/password encryption |
| 2018-12-11 | FK623Attend.dll(2.8.12.221) | Add JsonCommand interface to achieve SDK control device input data |
| FKAttend.dll(2.8.12.223) | Add 13\_1 communication protocol, JsonCommand universal interface to achieve command entry function (FK\_HS\_ExecJsonCmd) |
| FK623AttendDllCSSample.exe(2.3.8.5) | Add 13\_1 communication example, add a button example to realize the command entry function |
| datEnrollDat.mdb | The 32-bit string number was updated |

# 6.FAQs

## 6.1 How to Obtain All Information of All Users?

First, use ReadAllUserID to read IDs of all users and write them into the memory .Then, use

while Loop. 1. Use GetAllUserID to get ID of one user from memory. Then Some judgment conditions need to be added. Such as RUNERR\_DATAARRAY\_END, RUN\_SUCCESS .

2.Use GetEnrollData to obtain user information ,Card/password and fingerprint template , face template ,palm print template in byte array . When the above process is completed .Loop continue ,until all users enroll data are successfully acquired.

## 6.2 How to show Card/Passwords in the form of strings?

After Get byte array of card or password with cmd GetEnrollData，you can refer to Interface description above use the function in “Show card and password” to get string form of card/pwd

## 6.3 How to push User Card/Password to Machine?

you can refer to Interface description above,First use the function getPWD or getCard to return byte array.Then, use cmd “PutEnrollData” to push the card/pwd data.