# FK623Attend User`s Manual

( Version 2.8.8.6 )

2013.5

# Content

# 1 Introduction

**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. ocx, FK623Attend.dll and FKViaDev.dll for development of programs.**

**FK623Attend.ocx is an interface OCX for connection of the devices with the applications.**

**FK623Attend.dll is an interface DLL for connection of the devices with the applications. It has the same functions as FK623Attend.ocx.**

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

**The interface is composed of seven parts.**

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

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

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

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

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

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

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

# 2 FK623Attend.OCX Interface

## 2.1 Connection and Disconnection of Devices

ConnectComm

| | | |
|---|---|---|
| Type | long ConnectComm(<br>long anMachineNumber,<br>long anComPort,<br>long anBaudRate,<br>BSTR astrTelNumber,<br>long anWaitDialTime,<br>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 “4.2 Error Code Table”. | |
| Others | 1 | “pstrTelNumber” and “nWaitDialTime” are used when connecting to the device through the modem. Enter 0 when the modem is not used. |
| | 2 | “nLicense” is a license number granted to the device for the connection. Enter the correct license nuber, or it is unable to connect with the device. |

ConnectNet

| | | |
|---|---|---|
| Type | long ConnectNet(<br>long anMachineNumber,<br>BSTR astrIpAddress,<br>long anPort,<br>long anTimeOut,<br>long anProtocolType,<br>long anNetPassword,<br>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 “4.2 Error Code Table”. | |

| | | |
|---|---|---|
| Others | 1 | To return error codes after waiting as long as “nTimeOut” designates if the relevant device has not been connected to the network, |
| | 2 | “nProtocolType” designates the kind of protocol used for the network connection.<br>0 : PROTOCOL_TCPIP - TCP/IP communication<br>1 : PROTOCOL_UDP - UDP communication |
| | 3 | “nLicense” has the same meaning as “0 ConnectComm”. |

ConnectUSB

| | | |
|---|---|---|
| Type | long ConnectUSB(<br>long anMachineNumber,<br>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 “4.2 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 |

## 2.2 Management of Registered Data

GetEnrollData

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

| | | |
|---|---|---|
| | 2 | For the meanings of the operational authorization returned as "apnMachinePrivilege", please refer to "0 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".<br>The following values are returned to "anBackupNumber":<br>0 : BACKUP_FP_0 - registered in the first zone for fingerprints<br>… …<br>9 : BACKUP_FP_9 - registered in the ninth one<br>10 : BACKUP_PSW - passwords registered<br>11 : BACKUP_CARD – cards registered |

GetEnrollDataWithString

| | | |
|---|---|---|
| Type | long GetEnrollDataWithString(<br>long anEnrollNumber,<br>long anBackupNumber,<br>long* apnMachinePrivilege,<br>BSTR* apstrEnrollData) | |
| Functionality | To get the enrollment data in the type of a string. It is equal to GetEnrollData. | |
| Parameter | anEnrollNumber | Registration number |
| | anBackupNumber | Number classifying the kind of the enrollment data |
| | apnMachinePrivilege | Variable pointer of operational authorization of the registrants |
| | apstrEnrollData | Variable pointer of the enrollment datas |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | The enrollment data is alwayse returned to "apstrEnrollData" |
| | 2 | For other parameters, please refer to "0 GetEnrollData". |

PutEnrollData

| | | |
|---|---|---|
| Type | long PutEnrollData(<br>long anEnrollNumber,<br>long anBackupNumber,<br>long anMachinePrivilege,<br>long* apnEnrollData,<br>long anPassword) | |
| Functionality | To transmit to the device the enrollment data and operational authorization of the persons to be registereds | |
| Parameter | anEnrollNumber | Registration number |
| | anBackupNumber | Number classifying the kind of the enrollment data |
| | anMachinePrivilege | Operational authorization of the registrant |
| | apnEnrollData | Variable pointer of the fingerprint data |
| | anPassword | Password or card number data |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | As for "anBackupNumber", please refer to "0 GetEnrollData". |
| | 2 | As for "anMachinePrivilege", please refer to "0 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 "0 SaveEnrollData". |

PutEnrollDataWithString

| | | | |
|---|---|---|---|
| Type | long PutEnrollDataWithString(<br>long anEnrollNumber,<br>long anBackupNumber,<br>long anMachinePrivilege,<br>BSTR astrEnrollData) | | |
| Functionality | To contain the enrollment data in the type of a string. It is equal to PutEnrollData. | | |
| Parameter | anEnrollNumber | Registration number | |
| | anBackupNumber | Number classifying the kind of the enrollment data | |
| | anMachinePrivilege | Operational authorization of the registrants | |
| | astrEnrollData | Variable pointer of the enrollment data | |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | | |
| Others | 1 | The enrollment data are alwayse are contained by "apstrEnrollData". | |
| | 2 | As for the other parameters, please refer to "0 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 "4.2 Error Code Table". | |
| Others | 1 | Before using this command, you should transmit to the device the data to be registered with a command "PutEnrollData" or "PutEnrollDataWithString". |

DeleteEnrollData

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

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 "4.2 Error Code Table". | |

| | | |
|---|---|---|
| Others | 1 | The command fails to be executed when the structure of the file is not correct. |
| | 2 | To learn the method of using the USB memory in the device, please refer to the relevant user`s manual. |

USBReadAllEnrollDataCount

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

USBGetOneEnrollData

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

USBGetOneEnrollDataWithString

| | | |
|---|---|---|
| Type | long USBGetOneEnrollDataWithString(<br>long* apnEnrollNumber,<br>long* apnBackupNumber,<br>long* apnMachinePrivilege,<br>BSTR* apstrEnrollData,<br>long* apnEnableFlag,<br>BSTR* apstrEnrollName) | |
| Functionality | To get the enrollment data in the type of a string. It is equal to a "USBGetOneEnrollData". | |
| Parameter | apnEnrollNumber | Variable pointer of registration numbers |
| | apnBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| | apnMachinePrivilege | Variable pointer of the operational authorization of the registrants |
| | apstrEnrollData | Variable pointer of the enrollment data |
| | apnEnableFlag | Variable pointer of the flag enabling the registrant to use the device |
| | apnEnrollName | Variable pointer of the enroll name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | This command is similar to "GetEnrollDataWithString". The difference is that the former uses USB memories without connecting directly to the device. As for "GetEnrollDataWithString", please refer to "0 GetEnrollDataWithString". |
| | 2 | As for the others, please refer to "0 USBGetOneEnrollData". |

USBSetOneEnrollData

| | | |
|---|---|---|
| Type | long USBSetOneEnrollData(<br>long anEnrollNumber,<br>long anBackupNumber,<br>long anMachinePrivilege,<br>long* apnEnrollData,<br>long anPassWord,<br>long anEnableFlag,<br>BSTR anEnrollName) | |
| Functionality | To take a form in the internal memory of the PC in order to file the operational authorization and enrollment data of the person to be registered. The file can be used in USB memories. | |
| Parameter | anEnrollNumber | Registration number |
| | anBackupNumber | Number classifying the kind of the enrollment data |
| | anMachinePrivilege | Operational authorization of the registrant |
| | apnEnrollData | Variable pointer of the fingerprint data |
| | anPassWord | Password or card number data |
| | anEnableFlag | Flag enabling the registrant to use the device |
| | anEnrollName | Variable pointer of the enroll name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | This command is similar to "PutEnrollData". The difference is that the former uses USB memories without connecting directly to the device. As for "PutEnrollData", please refer to "0 PutEnrollData". |
| | 2 | For the meaning of "anEnableFlag", please refer to "0 EnableUser". |

USBSetOneEnrollDataWithString

| | | |
|---|---|---|
| Type | long USBSetOneEnrollDataWithString(<br>long anEnrollNumber,<br>long anBackupNumber,<br>long anMachinePrivilege,<br>BSTR astrEnrollData,<br>long anEnableFlag,<br>BSTR astrEnrollName) | |
| Functionality | To set the enrollment data in the type of string. This is equal to "USBSetOneEnrollD ata" | |
| Parameter | anEnrollNumber | Registration number |
| | anBackupNumber | Number classifying the kind of the enrollment data |
| | anMachinePrivilege | Operational authorization of the registrant |
| | astrEnrollData | Variable pointer of the enrollment data |
| | anEnableFlag | Flag enabling the registrant to use the device |
| | astrEnrollName | Variable pointer of the enroll name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | This command is similar to "PutEnrollDataWithString". The difference is that the former uses USB memories without connecting directly to the device. As for "PutEnrollDataWithString", please refer to "0 PutEnrollDataWithString". |
| | 2 | As for the others, please refer to "0 USBSetOneEnrollData". |

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

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 "4.2 Error Code Table". | |
| Others | 1 | The read information can be got with a command "GetAllUserID".<br>As for "GetAllUserID", please refer to "0 GetAllUserID". |
| | 2 | The command fails to be executed if the enrolled registrant does not exist. |

GetAllUserID

| | | |
|---|---|---|
| Type | long GetAllUserID(<br>long* apnEnrollNumber,<br>long* apnBackupNumber,<br>long* apnMachinePrivilege,<br>long* apnEnableFlag) | |
| Functionality | To get one by one the registrants` information read with “ReadAllUserID”. | |
| Parameter | apnEnrollNumber | Variable pointer of the registration number |
| | apnBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| | apnMachinePrivilege | Variable pointer of the operational authorization of the registrant |
| | apnEnableFlag | Variable pointer of the flag enabling the registrant to use the device |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”. | |
| Others | 1 | The command fails to be executed if there is no registrant`s information read by “ReadAllUserID”. |
| | 2 | Code “RUNERR_LOG_END” is returned after the data are all got. |
| | 3 | For the meaning of the operational authorization returned with “apnMachinePrivilege”, please refer to “0 ModifyPrivilege”. |
| | 4 | For the meaning of “apnEnableFlag”, please refer to “0 EnableUser”. |

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 “4.2 Error Code Table”. | |
| Others | 1 | Before the execution of this command, it is necessary to backup the registered data. |

ClearKeeperData

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

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

GetVerifyMode

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

SetVerifyMode

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

USBGetOneEnrollData_1

| | | |
|---|---|---|
| Type | long USBGetOneEnrollData_1(<br>long* apnEnrollNumber,<br>long* apnBackupNumber,<br>long* apnVerifyMode,<br>long* apnMachinePrivilege,<br>long* apnEnrollData,<br>long* apnPassWord,<br>long* apnEnableFlag,<br>BSTR* apstrEnrollName) | |
| Functionality | To get the enrollment data read with a command “USBReadAllEnrollDataFromFile”. | |
| Parameter | apnEnrollNumber | Variable pointer of registration numbers |
| | apnBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| | apnVerifyMode | Variable pointer of verify mode of the users |
| | apnMachinePrivilege | Variable pointer of the operational authorization of the registrants |
| | apnEnrollData | Variable pointer of the fingerprint data |
| | apnPassWord | Variable pointer of the password or card number data |
| | apnEnableFlag | Variable pointer of the flag enabling the registrant to use the device |
| | apstrEnrollName | Variable pointer of the enroll name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”. | |
| Others | 1 | This command is similar to “GetEnrollData”. The difference is that the former uses USB memories without connecting directly to the device. For the description of “GetEnrollData”, please refer to “0 GetEnrollData”. |
| | 2 | To return a code “RUNERR_LOG_END” after getting all the data |
| | 3 | The command fails to be executed when there is no enrollment data read into the PC with a command “USBReadAllEnrollDataFromFile”. |
| | 4 | For the meaning of “apnEnableFlag”, please refer to “0 EnableUser”. |

USBGetOneEnrollDataWithString_1

| | | |
|---|---|---|
| Type | long USBGetOneEnrollDataWithString_1(<br>    long* apnEnrollNumber,<br>    long* apnBackupNumber,<br>    long* apnVerifyMode,<br>    long* apnMachinePrivilege,<br>    BSTR* apstrEnrollData,<br>    long* apnEnableFlag,<br>    BSTR* apnEnrollName) | |
| Functionality | To get the enrollment data in the type of a string. It is equal to a “USBGetOneEnrollData”. | |
| Parameter | apnEnrollNumber | Variable pointer of registration numbers |
| | apnBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| | apnVerifyMode | Variable pointer of verify mode of the users |
| | apnMachinePrivilege | Variable pointer of the operational authorization of the registrants |
| | apstrEnrollData | Variable pointer of the enrollment data |
| | apnEnableFlag | Variable pointer of the flag enabling the registrant to use the device |
| | apnEnrollName | Variable pointer of the enroll name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”. | |
| Others | 1 | This command is similar to “GetEnrollDataWithString”. The difference is that the former uses USB memories without connecting directly to the device. As for “GetEnrollDataWithString”, please refer to “0 GetEnrollDataWithString”. |
| | 2 | As for the others, please refer to “0 USBGetOneEnrollData”. |

USBSetOneEnrollData_1

| | | |
|---|---|---|
| Type | long USBSetOneEnrollData_1(<br>    long anEnrollNumber,<br>    long anBackupNumber,<br>    long anVerifyMode,<br>    long anMachinePrivilege,<br>    long* apnEnrollData,<br>    long anPassWord,<br>    long anEnableFlag,<br>    BSTR astrEnrollName) | |
| Functionality | To take a form in the internal memory of the PC in order to file the operational authorization and enrollment data of the person to be registered. The file can be used in USB memories. | |
| Parameter | anEnrollNumber | Registration number |
| | anBackupNumber | Number classifying the kind of the enrollment data |
| | anVerifyMode | Variable pointer of verify mode of the users |
| | anMachinePrivilege | Operational authorization of the registrant |
| | apnEnrollData | Variable pointer of the fingerprint data |
| | anPassWord | Password or card number data |
| | anEnableFlag | Flag enabling the registrant to use the device |
| | astrEnrollName | Variable pointer of the enroll name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”. | |
| Others | 1 | This command is similar to “PutEnrollData”. The difference is that the former uses USB memories without connecting directly to the device. As for “PutEnrollData”, please refer to “0 PutEnrollData”. |
| | 2 | For the meaning of “anEnableFlag”, please refer to “0 EnableUser”. |

USBSetOneEnrollDataWithString_1

| | | |
|---|---|---|
| Type | long USBSetOneEnrollDataWithString_1(<br>long anEnrollNumber,<br>long anBackupNumber,<br>long anVerifyMode,<br>long anMachinePrivilege,<br>BSTR astrEnrollData,<br>long anEnableFlag,<br>BSTR astrEnrollName) | |
| Functionality | To set the enrollment data in the type of string. This is equal to "USBSetOneEnrollD ata" | |
| Parameter | anEnrollNumber | Registration number |
| | anBackupNumber | Number classifying the kind of the enrollment data |
| | anVerifyMode | Variable of verify mode of the users |
| | anMachinePrivilege | Operational authorization of the registrant |
| | astrEnrollData | Variable pointer of the enrollment data |
| | anEnableFlag | Flag enabling the registrant to use the device |
| | astrEnrollName | Variable pointer of the enroll name |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | This command is similar to "PutEnrollDataWithString". The difference is that the former uses USB memories without connecting directly to the device. As for "PutEnrollDataWithString", please refer to "0 PutEnrollDataWithString". |
| | 2 | As for the others, please refer to "0 USBSetOneEnrollData". |

USBReadAllEnrollDataFromFile_Color

| | | |
|---|---|---|
| Type | long USBReadAllEnrollDataFromFile_Color(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 "4.2 Error Code Table". | |
| Others | 1 | The command fails to be executed when the structure of the file is not correct. |
| | 2 | To learn the method of using the USB memory in the device, please refer to the relevant user`s manual. |

USBWriteAllEnrollDataToFile_Color

| | | |
|---|---|---|
| Type | long USBWriteAllEnrollDataToFile_Color(BSTR astrFilePath, long anNewsKind) | |
| Functionality | To file the enrollment data formed in the internal memory of the PC by "USBSetOneEnrollData" or "USBSetOneEnrollDataWithString" | |
| Parameter | astrFilePath | File name |
| | anNewsKind | News Kind : NewKind = 0x02 : 60 chineses characters<br>NewKind = 0x01 : 24 chineses characters |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | Before the execution of the command, there should be the data formed by the command "USBSetOneEnrollData" or "USBSetOneEnrollDataWith String". |

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

USBGetOneEnrollData_Color

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

USBGetOneEnrollDataWithString_Color

| | | |
|---|---|---|
| Type | long USBGetOneEnrollDataWithString_Color(<br>long* apnEnrollNumber,<br>long* apnBackupNumber,<br>long* apnMachinePrivilege,<br>BSTR* apstrEnrollData,<br>long* apnEnableFlag,<br>BSTR* apstrEnrollName,<br>long anNewsKind) | |
| Functionality | To get the enrollment data in the type of a string. It is equal to a "USBGetOneEnrollData". | |
| Parameter | apnEnrollNumber | Variable pointer of registration numbers |
| | apnBackupNumber | Variable pointer of number classifying the kind of enrollment data |
| | apnMachinePrivilege | Variable pointer of the operational authorization of the registrants |
| | apstrEnrollData | Variable pointer of the enrollment data |
| | apnEnableFlag | Variable pointer of the flag enabling the registrant to use the device |

| | | | |
|---|---|---|---|
| | apstrEnrollName | Variable pointer of the enroll name | |
| | anNewsKind | News Kind : NewKind = 0x02 : 60 chineses characters<br>NewKind = 0x01 : 24 chineses characters | |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | | |
| Others | 1 | This command is similar to "GetEnrollDataWithString". The difference is that the former uses USB memories without connecting directly to the device. As for "GetEnrollDataWithString", please refer to "0 GetEnrollDataWithString". | |
| | 2 | As for the others, please refer to "0 USBGetOneEnrollData". | |

USBSetOneEnrollData_Color

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

USBSetOneEnrollDataWithString_Color

| | | |
|---|---|---|
| Type | long USBSetOneEnrollDataWithString_Color(<br>long anEnrollNumber,<br>long anBackupNumber,<br>long anMachinePrivilege,<br>BSTR astrEnrollData,<br>long anEnableFlag,<br>BSTR anEnrollName,<br>long anNewsKind) | |
| Functionality | To set the enrollment data in the type of string. This is equal to "USBSetOneEnrollD ata" | |
| Parameter | anEnrollNumber | Registration number |

| | | |
|---|---|---|
| | anBackupNumber | Number classifying the kind of the enrollment data |
| | anMachinePrivilege | Operational authorization of the registrant |
| | apstrEnrollData | Variable pointer of the enrollment data |
| | anEnableFlag | Flag enabling the registrant to use the device |
| | anEnrollName | Variable pointer of the enroll name |
| | anNewsKind | News Kind : NewKind = 0x02 : 60 chineses characters<br>NewKind = 0x01 : 24 chineses characters |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”. | |
| Others | 1 | This command is similar to “PutEnrollDataWithString”. The difference is that the former uses USB memories without connecting directly to the device. As for “PutEnrollDataWithString”, please refer to “0 PutEnrollDataWithString”. |
| | 2 | As for the others, please refer to “0 USBSetOneEnrollData”. |

## 2.3 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 “4.2 Error Code Table”. | |
| Others | 1 | The read data can be got by “GetSuperLogData” Please refer to “0<br><br>GetSuperLogData”. |
| | 2 | anReadMark = 1 permits reading the newly-added recorded data alone.<br>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 “4.2 Error Code Table”. | |
| Others | 1 | Similar to LoadSuperLogData, this command can be used to get the administrative data when the device has not been connected with the PC. |
| | 2 | The incorrect structures of the files result in a failure of the execution. |
| | 3 | For the method of using USB memories in the devices, please refer to the corresponding user`s manual. |

GetSuperLogData

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

| | | |
|---|---|---|
| Others | 1 | The read data can be got by "GetGeneralLogData".<br>Please, refer to "0 GetGeneralLogData". |
| | 2 | anReadMark = 1 allows to read newly-added recorded data alone.<br>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 "4.2 Error Code Table". | |
| Others | 1 | Similar to "LoadGeneralLogData", this command can be used to get the attendance data when the device is not connected with the PC. |
| | 2 | The incorrect structure of the file results in a failure of the execution. |
| | 3 | For the method of using USB memories in the devices, please refer to the corresponding user`s manual. |

GetGeneralLogData

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

| | | |
|---|---|---|
| | 2 | The following values are returned to "apnVerifyMode":<br>1 : LOG_FPVERIFY - Verified as fingerprints<br>2 : LOG_PASSVERIFY - Verified as passwords<br>3 : LOG_CARDVERIFY - Verified as cards<br>4 : LOG_FPPASS_VERIFY - Verified as passwords added to fingerprints<br>5 : LOG_FPCARD_VERIFY - Verified as cards added to fingerprints<br>6 : LOG_PASSFP_VERIFY - Verified as fingerprints added to passwords<br>7 : LOG_CARDFP_VERIFY - Verified as fingerprints added to cards<br>(The followings are used in the models with a function of controlling doors. Refer to "2.7 Control of Doors".)<br>10 : LOG_OPEN_DOOR - The signal of opening the door is transmitted after the verification.<br>11 : LOG_CLOSE_DOOR - The signal of closing the door is transmitted after the verification<br>12 : LOG_OPEN_HAND - The signal of opening the door with the key is transferred.<br>13 : LOG_OPEN_THREAT - The signal of opening the door by verifying threatened fingerprints is transferred.<br>14 : LOG_PROG_OPEN - The signal of opening the door is transferred from the controlling device.<br>15 : LOG_PROG_CLOSE - The signal of closing the door is transferred from the controlling device.<br>16 : LOG_OPEN_IREGAL - The signal of opening the door is illegally transferred.<br>17 : LOG_CLOSE_IREGAL - The signal of closing the door is illegally transferred.<br>18 : LOG_OPEN_COVER - The cover of the device opened<br>19 : LOG_CLOSE_COVER - The cover of the device closed |
| | 3 | This command fails to be executed unless "LoadGeneralLogData" or "USBLoadGeneralLogDataFromFile" is first executed. |
| | 4 | The following values are returned to "apnInOutMode":<br>0 : LOG_IOMODE_IN - Verified with the mode of coming in<br>1 : LOG_IOMODE_OUT - Verified with the mode of going out<br>2 : 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 "4.2 Error Code Table". | |
| Others | 1 | It is necessary to backup the data relating to incoming and outgoing before the execution of this command. |

GetGeneralLogData_1

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

| | | |
|---|---|---|
| | 3 | This command fails to be executed unless "LoadGeneralLogData" or "USBLoadGeneralLogDataFromFile" is first executed. |
| | 4 | The following values are returned to "apnInOutMode":<br>0 : LOG_IOMODE_IN - Verified with the mode of coming in<br>1 : LOG_IOMODE_OUT - Verified with the mode of going out<br>2 : LOG_IOMODE_IO - Verified with the general mode |

GetSuperLogData_1

| | | |
|---|---|---|
| Type | long GetSuperLogData_1(<br>long* apnSEnrollNumber,<br>long* apnGEnrollNumber,<br>long* apnManipulation,<br>long* apnBackupNumber,<br>long* apnYear,<br>long* apnMonth,<br>long* apnDay,<br>long* apnHour,<br>long* apnMinute,<br>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<br>apnDay, apnHour<br>apnMinute, apnSec | Variable pointer of the time and the date when the management was recorded |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | After all the data are got, a code "RUNERR_LOG_END" is return. |
| | 2 | This command fails to be executed if "LoadSuperLogData" or "USBLoadSuperLogDataFromFile" is not first executed. |

| | | |
|---|---|---|
| | 3 | The following values are returned to "apnManipulation":<br>3 : LOG_ENROLL_USER - To register general users<br>4 : LOG_ENROLL_MANAGER - To register manager(s)<br>5 : LOG_ENROLL_DELFP - To delete fingerprint data<br>6 : LOG_ENROLL_DELPASS - To delete passwords<br>7 : LOG_ENROLL_DELCARD - To delete card data<br>8 : LOG_LOG_ALLDEL - To delete all the management data<br>9 : LOG_SETUP_SYS - To modify the information about the devices<br>10 : LOG_SETUP_TIME - To modify the time of the devices<br>11 : LOG_SETUP_LOG - To modify the limit values of the management data<br>12 : LOG_SETUP_COMM - To modify the communication modes<br>13 : LOG_PASSTIME - To set the duration for which the doors are passed through<br>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 "4.2 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 "4.2 Error Code Table". | |

## 2.4 Management of Registrants` Information

EnableUser

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

ModifyPrivilege

| | | |
|---|---|---|
| Type | long ModifyPrivilege(<br>long anEnrollNumber,<br>long anBackupNumber,<br>long anMachinePrivilege) | |
| Functionality | To set the operational authorization of the registrant | |
| Parameter | anEnrollNumber | Registration number |
| | anBackupNumber | Number classifying the kind of the enrollment data |
| | anMachinePrivilege | Operational authorization |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | The registrants can be divided into managers and general users according to the operational authorization. This authorization is reflected in "anMachinePrivilege".<br>The following values are returned to "anMachinePrivilege":<br>0 : MP_NONE - General user (can only be verified through the device.)<br>1 : 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 "4.2 Error Code Table". | |
| Others | 1 | The maximum size of the name contained by "apstrUserName" is 10byte (10 English letters or 5 other letters at most). |
| | 2 | The command fails to be executed if no name is assigned. |

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 "4.2 Error Code Table". | |
| Others | 1 | The maximum size of the name contained by "apstrUserName" is 10byte (10 English letters or 5 other letters at most). |
| | 2 | The command fails to be executed if no name is assigned. |

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 "4.2 Error Code Table". | |
| Others | 1 | "anNewsId" is a number designating messages. The range is from 0 upto 255. |

| | | |
|---|---|---|
| | 2 | The maximum size of the name contained by "apstrUserName" is 48byte (48 English letters or 24 other letters at most). |

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 "4.2 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 "4.2 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 "4.2 Error Code Table". | |
| Others | 1 | "apnNewsId" is a value to be set under "0 SetNewsMessage". |

## 2.5 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 "4.2 Error Code Table". | |
| Others | 1 | It can be used when forbidding the operation on the device for the communication between the PC and the device. |
| | 2 | anEnabledFlag=0 forbids the operation with a message "Working…" prompted; anEnabledFlag=1 allows it with the normal display shown. |

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 “4.2 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 “4.2 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 “4.2 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 “4.2 Error Code Table”. | |
| Others | 1 | This command helps seize the current status of the device through the PC. |
| | 2 | The following values are returned to “anStatusIndex”:<br>1 : GET_MANAGERS - The number of managers existing currently<br>2 : GET_USERS - The number of general users existing currently<br>3 : GET_FPS - The number of fingerprint data existing currently<br>4 : GET_PSWS - The number of password data existing currently<br>5 : GET_SLOGS - The number of new management data existing currently<br>6 : GET_GLOGS - The number of new Income/Outgoing existing-data.<br>7 : GET_ASLOGS - The number of the entire management existing –data.<br>8 : GET_AGLOGS - The number of the entire Income/Outgoing existing-data.<br>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 "4.2 Error Code Table". | |
| Others | 1 | The following values are returned to "anInfoIndex":<br>1 : DI_MANAGERS - The maximum number of registerable managers<br>2 : DI_MACHINENUM - ID number of the device<br>3 : DI_LANGAUGE - Language displayed on the device<br>4 : DI_POWEROFF_TIME - Auto-poweroff duration<br>5 : DI_LOCK_CTRL - Door control flag<br>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.<br>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<br>8 : DI_VERIFY_INTERVALS- Interval for recording verification. Within this time, the repeated verification is not recorded.<br>9 : DI_RSCOM_BPS – Baudrate of the serial communication<br>Each of the baudrates has the following value.<br>BPS_9600 = 3<br>BPS_19200 = 4<br>BPS_38400 = 5<br>BPS_57600 = 6<br>BPS_115200 = 7<br>10: DI_DATE_SEPARATE- Type of displaying time and dates<br>11: DI_VERIFY_KIND: setting of matching modes<br>the setting values for matching modes are the followings.<br>0: F / P / C<br>1: F + P<br>2: F + C<br>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 "4.2 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 “4.2 Error Code Table”. | | |
| Others | 1 | The following values are returned to “anProductIndex”:<br>1 : PRODUCT_SERIALNUMBER - Serial number<br>2 : PRODUCT_BACKUPNUMBER - Subscription number<br>3 : PRODUCT_CODE - Model number<br>4 : PRODUCT_NAME - Model name<br>5 : PRODUCT_WEB - Homepage of the seller<br>6 : PRODUCT_DATE - Sale date<br>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 “4.2 Error Code Table”. | |
| Others | 1 | |

GetDeviceTime_1

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

SetDeviceTime_1

| | |
|---|---|
| Type | long SetDeviceTime_1(<br>long anYear,<br>long anMonth,<br>long anDay,<br>long anHour,<br>long anMinute,<br>long anSec,<br>long anDayOfWeek) |

| | | |
|---|---|---|
| Functionality | To set time and a date on the device | |
| Parameter | apnYear,apnMonth<br>apnDay, apnHour<br>apnMinute, apnSec<br>anDayOfWeek | Time and date data |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | |

## 2.6 Management of Bells

GetBellTime

| | | |
|---|---|---|
| Type | long GetBellTime(long* apnBellCount, long* aptBellInfo) | |
| Functionality | To get the information about setting a bell | |
| Parameter | apnBellCount | Variable pointer of times of the bell ringing |
| | aptBellInfo | Variable pointer of the bell information structure |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | The number of bells ringing at the same time is returned to "apnBellCount". |
| | 2 | The information about the bell such as the designated number and time is returned to "aptBellInfo". For the meaning, please refer to "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 "4.2 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 "4.2 Error Code Table". | |
| Others | 1 | The number of bells ringing at the same time is returned to "apnBellCount". |
| | 2 | The information about the bell such as the designated number and time is returned to "aptBellInfo". For the meaning, please refer to "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 "4.2 Error Code Table". | |
| Others | 1 | For the details, please refer to "0 SetBellTime". |

## 2.7 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 "4.2 Error Code Table". | |
| Others | 1 | The following values are returned to "apnStatusVal":<br>0 : DOOR_CONTROLRESET - control state of door by device.<br>1 : DOOR_OPEND - Door opened<br>2 : DOOR_CLOSED – Door closed<br>3 : DOOR_COMMNAD- by the command for control of doors, door opend for some time and closed. |

SetDoorStatus

| | | |
|---|---|---|
| Type | long SetDoorStatus(long anStatusVal) | |
| Functionality | To control the door opening status | |
| Parameter | anStatusVal | Status value |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | for the meanings of "anStatusVal", refer to "0 GetDoorStatus". |

GetPassTime

| | | |
|---|---|---|
| Type | long GetPassTime(long anPassTimeID, long* apnPassTime, long anPassTimeSize) | |
| Functionality | To get the information about the time zone of opening or closing the door | |
| Parameter | anPassTimeID | ID number of the information about the time zone |
| | apnPassTime | Variable pointer of the structure of the above information |
| | anPassTimeSize | Length of the above structure |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | "anPassTimeID" is a number indicating the structure of the information about the time zone.<br>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.<br>Please refer to "0<br><br>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 "4.2 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 "4.2 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 "4.2 Error Code Table". | |
| Others | 1 | For the details, please refer to "0 GetPassTime". |

GetUserPassTime

| | | |
|---|---|---|
| Type | long GetUserPassTime(<br>long anEnrollNumber,<br>long *apnGroupID,<br>long *apnPassTimeID,<br>long anPassTimeIDSize) | |
| Functionality | To get the time zone-relaing information group assigned to the designated user and the group assigned individually | |
| Parameter | anEnrollNumber | Registration number |
| | apnGroupID | Variable pointer of group number |

| | | |
|---|---|---|
| | apnPassTimeID | Variable pointer of the structure of the ID number for the information about the time zone |
| | anPassTimeIDSize | Length of the above structure |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | For the meaning of "apnGroupID", please refer to "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(<br>long anEnrollNumber,<br>long* apnGroupID,<br>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 "4.2 Error Code Table". | |
| Others | 1 | For the details, please refer to "0 GetUserPassTime". |

SetUserPassTime

| | | |
|---|---|---|
| Type | long SetUserPassTime(<br>long anEnrollNumber,<br>long anGroupID,<br>long* apnPassTimeID,<br>long anPassTimeIDSize) | |
| Functionality | To set the information group of the time zone and the individually-assigned information for the designated registrant | |
| Parameter | anEnrollNumber | Registration number |
| | anGroupID | Group number |
| | apnPassTimeID | Variable pointer of the ID number structure of the time zone information |
| | anPassTimeIDSize | Length of the above structure |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | For the details, please refer to "0 GetUserPassTime". |

SetUserPassTimeWithString

| | |
|---|---|
| Type | long SetUserPassTimeWithString(<br>long anEnrollNumber,<br>long anGroupID,<br>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 "4.2 Error Code Table". | |
| Others | 1 | For the details, please refer to "0 GetUserPassTime". |

GetGroupPassTime

| | | |
|---|---|---|
| Type | long GetGroupPassTime(<br>long anGroupID,<br>long *apnPassTimeID,<br>long anPassTimeIDSize) | |
| Functionality | To get ID numbers of the time zone information corresponding to the designated time zone information group | |
| Parameter | anGroupID | Group number |
| | apnPassTimeID | Variable pointer of the ID number structure for the time zone information |
| | anPassTimeIDSize | Length of the above structure |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to "4.2 Error Code Table". | |
| Others | 1 | On the devices, structures of time zone information can be used in groups. "anGroupID" is a number indicating the group.<br>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(<br>long anGroupID,<br>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 "4.2 Error Code Table". | |
| Others | 1 | For the details, please refer to "0 GetGroupPassTime". |

SetGroupPassTime

| | | |
|---|---|---|
| Type | long SetGroupPassTime(<br>long anGroupID,<br>long *apnPassTimeID,<br>long anPassTimeIDSize) | |
| Functionality | To set ID numbers of the time zone information in the designated group of the information | |
| Parameter | anGroupID | Group number |
| | apnPassTimeID | Variable pointer of the ID number structure of the time zone information |
| | anPassTimeIDSize | Length of the above structure |
| Return | Success returns 1; failure returns the corresponding error code. For details of error codes, please refer to “4.2 Error Code Table”. | |
| Others | 1 | For the details, please refer to “0 GetGroupPassTime”. |

SetGroupPassTimeWithString

| | | |
|---|---|---|
| Type | long SetGroupPassTimeWithString(<br>long anGroupID,<br>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 “4.2 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 “4.2 Error Code Table”. | |
| Others | 1 | To combine the groups of the time zone information structures and use them for control of the doors (opening or closing doors)<br>Ten unions at most can be formed. “apnGroupMatch” is an array-typed batch structure for these unions.<br>For the definition of the structure, please refer to “0 GROUPMATCHINFO Structure”.<br>Group numbers are described one after another on the item of structures<br>Ex: ‘13’ described if groups No.1 and No.3 are combined at the same time,<br>‘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 “4.2 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 “4.2 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 “4.2 Error Code Table”. | |
| Others | 1 | For the details, please refer to “0 GetGroupMatch”. |

## 2.8 Adjust Management

GetAdjustInfo

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

SetAdjustInfo

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

## 2.9 Network Information Management

GetServerNetInfo

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

SetServerNetInfo

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

SetUSBModel

| | |
|---|---|
| Type | void SetUSBModel(long anModel) |

| | | |
|---|---|---|
| Functionality | To set machine model info for USB Flash information | |
| others | 1 | "anModel" is a machine model info. |

## 2.10 Post & Shift Management

GetOneShiftInfo

| | | |
|---|---|---|
| Type | long GetOneShiftInfo(<br>long anShiftNumber,<br>long* apShiftSHour,<br>long* apShiftSMinute,<br>long* apShiftEHour,<br>long* apShiftEMinute,<br>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 "4.2 Error Code Table". | |

SetOneShiftInfo

| | | |
|---|---|---|
| Type | long SetOneShiftInfo(<br>long anShiftNumber,<br>long anShiftSHour,<br>long anShiftSMinute,<br>long anShiftEHour,<br>long anShiftEMinute,<br>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 "4.2 Error Code Table". | |

GetOnePostInfo

| | | |
|---|---|---|
| Type | long GetOnePostInfo(<br>long anPostNumber,<br>BSTR* apStrPostName,<br>long* apShiftNumber1,<br>long* apShiftNumber2,<br>long* apShiftNumber3,<br>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 "4.2 Error Code Table". | |

SetOnePostInfo

| | | |
|---|---|---|
| Type | long SetOnePostInfo(<br>long anPostNumber,<br>BSTR astrPostName,<br>long anShiftNumber1,<br>long anShiftNumber2,<br>long anShiftNumber3,<br>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 "4.2 Error Code Table". | |

GetUserInfo

| | |
|---|---|
| Type | long GetUserInfo(<br>long anEnrollNumber,<br>BSTR* apstrUserName,<br>long* apNewKind,<br>long* apVerifyMode,<br>long* apPostID,<br>long* apShiftNumber1,<br>long* apShiftNumber2,<br>long* apShiftNumber3,<br>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 “4.2 Error Code Table”. | |

SetUserInfo

| | | |
|---|---|---|
| Type | long SetUserInfo(<br>long anEnrollNumber,<br>BSTR astrUserName,<br>long anNewKind,<br>long anVerifyMode,<br>long anPostID,<br>long anShiftNumber1,<br>long anShiftNumber2,<br>long anShiftNumber3,<br>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 “4.2 Error Code Table”. | |

# 3 FK623Attend.DLL Interface

The interfaces of FK623Attend.DLL are similar to the ones of FK623Attend.OCX.

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

## 3.1 Differences in interface

First, the DLL interface functions have “FK_”prefix.

For example :

| interface function of OCX | interface function of DLL |
|---|---|
| long ConnectNet(<br>long nMachineNumber,<br>BSTR strIpAddress,<br>long nPort,<br>long nTimeOut,<br>long nProtocolType,<br>long nNetPassword,<br>long nLicense); | long FK_ConnectNet(<br>long anMachineNo,<br>const char* astrIpAddress,<br>long anNetPort,<br>long anTimeOut,<br>long anProtocolType,<br>long anNetPassword,<br>long anLicense); |

Second, there are differences in parameter declaration.

The DLL interface functions have one additional parameter of type long that indicate connection handle in addition to the corresponding OCX interface function. This additional parameter is the first parameter of most DLL interface functions.

For example :

| interface function of OCX | interface function of DLL |
|---|---|
| long GetEnrollData(<br>long anEnrollNumber,<br>long anBackupNumber,<br>long* apnMachinePrivilege,<br>long* apnEnrollData,<br>long* apnPassWord); | long FK_GetEnrollData(<br>long anHandleIndex,<br>long anEnrollNumber,<br>long anBackupNumber,<br>long* apnMachinePrivilege,<br>void* apEnrollData,<br>long* apnPassWord); |

Third, there are differences in the code of string argument of interface functions.

The string arguments passed to OCX functions are based on Unicode(16bit) but the string arguments passed to DLL functions are based on ANSI code according to the language setting on Windows system .

## 3.2 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 OCX | interface function of DLL |
|---|---|
| long GetUserName(<br>    long anEnrollNumber,<br>    BSTR* apstrUserName); | long FK_GetUserName(<br>    long anHandleIndex,<br>    long anEnrollNumber,<br>    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

```
//-- 函数宣言部 ----------------------------------------------------
long FP_EXPORT FK_GetUserName(long anHandleIndex, long anEnrollNumber, char** apstrUserName);
// ------------------------------------------------------------------


//-- 例子代码 -------------------------------------------------------
char*     pszTemp = new char[256];
nErrorCode = FK_GetUserName(m_nCommHandleIndex, nEnrollNumber, &pszTemp);
AfxMessageBox(pszTemp);
delete [] pszTemp;
```

# 4 Appendix

## 4.1 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.
```

## 4.2 Error Code Table

| Value | Symbol | Description |
|---|---|---|
| 1 | RUN_SUCCESS | Message informing of the successful execution of commands |
| 0 | RUNERR_NOSUPPORT | Error that the device does not support the relevant command |
| -1 | RUNERR_UNKNOWNERROR | Unknown error |
| -2 | RUNERR_NO_OPEN_COMM | Error that the device has been not connected to |
| -3 | RUNERR_WRITE_FAIL | Error that the data has not been transmitted to the device |
| -4 | RUNERR_READ_FAIL | Error that the data has not been read from the device |
| -5 | RUNERR_INVALID_PARAM | Error that the input parameters are not correct |
| -6 | RUNERR_NON_CARRYOUT | Error that the command has not been executed correctly |
| -7 | RUNERR_DATAARRAY_END | Message telling that there is no more data to get |
| -8 | RUNERR_DATAARRAY_NONE | Error that the data do not exist |
| -9 | RUNERR_MEMORY | Error that the memory of the PC is not enough |
| -10 | RUNERR_MIS_PASSWORD | Error that the input license does not accord when connecting with the device |
| -11 | RUNERR_MEMORYOVER | Error that the memory has no space where more enrollment data can be registered in the device |
| -12 | RUNERR_DATADOUBLE | Error that the registration number to be enrolled is already stored in the database of the device |
| -14 | RUNERR_MANAGEROVER | Error that the memory has no space where more data of the manager can be registered in the device |
| -15 | RUNERR_FPDATAVERSION | Error that the version of the fingerprint data to be used is not correct |