

# **iFace Series**

## **Communication Protocol SDK**

### **Development Handbook**

---

**Date: July, 2012**

2012 ZKSoftware Inc. All rights reserved.

If there is some change of information in this handbook, ZKSoftware won't inform specially.

- ZKSoftware Inc. is our company logo. Biokey is core technology logo of ZKSoftware. Both logos have been registered in China and America.
- Other trademarks and product names mentioned in this handbook are produced by other companies. ZKSoftware Inc. has no property of them.
- Please solve use problems (instead of development kit problem) during development by yourself.
- ZKSoftware has no responsibility for data loss caused by users or programs.
- This handbook is designed on the basis of SDK 6.2.4.1.

# Contents

<b>Communication Protocol SDK.....</b>	<b>1</b>
<b>Development Handbook .....</b>	<b>1</b>
<b>Contents .....</b>	<b>3</b>
<b>1 SDK Description.....</b>	<b>7</b>
<b>2 Quick Start.....</b>	<b>7</b>
2.1 Terms .....	7
2.2 Common Processes.....	9
2.2.1 Downloading Attendance Records .....	9
2.2.2 Downloading Operation Records.....	9
2.2.3 Setting Access Control.....	10
2.2.4 Downloading User Information, Fingerprint Templates, or Face Templates.....	10
2.2.5 Receiving Real-time Events .....	12
2.2.6 Enrolling Users Online (Uploading Information, Fingerprint Templates, and Face Templates of Users) .....	12
<b>3 Related Attributes .....</b>	<b>13</b>
3.1 AccGroup.....	13
3.2 AccTimeZones.....	13
3.3 BASE64.....	14
3.4 CardNumber .....	14
3.5 CommPort .....	14
3.6 ConvertBIG5.....	14
3.7 PINWidth .....	14
3.8 GetStrCardNumber .....	14
3.9 SetStrCardNumber .....	15
<b>4 Real-time Event Functions.....</b>	<b>15</b>
4.1 Obtaining Real-Time Events.....	15
4.1.1 RegEvent.....	15
4.1.2 ReadRTLog .....	16
4.1.3 GetRTLog.....	16
4.2 Real-Time Events .....	17
4.2.1 OnConnected .....	17
4.2.2 OnDisConnected .....	17
4.2.3 OnAlarm .....	17
4.2.4 OnDoor.....	17
4.2.5 OnAttTransactionEx.....	18

4.2.6 OnEnrollFingerEx .....	19
4.2.7 OnFinger .....	19
4.2.8 OnFingerFeature .....	19
4.2.9 OnHIDNum.....	19
4.2.10 OnNewUser .....	19
4.2.11 OnVerify .....	19
4.2.12 OnWriteCard.....	20
4.2.13 OnEmptyCard.....	20
4.2.14 OnEMData.....	20
<b>5 Common Functions.....</b>	<b>20</b>
5.1 Device Connection Functions.....	20
5.1.1 Connect_Net.....	20
5.1.2 Connect_Com.....	21
5.1.3 Disconnect.....	21
5.2 Data Management Functions .....	22
5.2.1 Attendance Record Data.....	22
5.2.2 Operation Record Data .....	24
5.2.3 User Information Functions .....	29
5.2.4 Enroll Data Functions (Including Both User Information and Fingerprint) .....	34
5.2.5 Fingerprint Template Functions.....	35
5.2.6 Face Data Functions.....	40
5.2.7 Fingerprint Template Conversion Functions .....	43
5.2.8 System Data Management Functions.....	46
5.3 Access Control Functions (Time Zone, Group, Open Door Combination).....	49
5.3.1 GetUserGroup .....	49
5.3.2 SetUserGroup.....	49
5.3.3 GetTZInfo .....	50
5.3.4 SetTZInfo.....	50
5.3.5 SSR_SetUnLockGroup .....	51
5.3.6 SSR_GetUnLockGroup.....	52
5.3.7 SSR_SetGroupTZ.....	52
5.3.8 SSR_GetGroupTZ .....	53
5.3.9 GetUserTZs.....	54
5.3.10 SetUserTZs .....	54
5.3.11 GetUserTZStr .....	55
5.3.12 SetUserTZStr.....	55
5.3.13 ACUnlock .....	56
5.3.14 GetACFun.....	56
5.3.15 GetDoorState.....	57
5.3.16 UseGroupTimeZone .....	57
5.4 Device Management Functions .....	58
5.4.1 IsTFTMachine.....	58
5.4.2 GetDeviceStatus.....	58
5.4.3 GetDeviceInfo.....	59

5.4.4 SetDeviceInfo .....	62
5.4.5 SetDeviceTime .....	62
5.4.6 SetDeviceTime2 .....	63
5.4.7 GetDeviceTime.....	63
5.4.8 GetSerialNumber .....	64
5.4.9 GetProductCode .....	64
5.4.10 GetFirmwareVersion .....	65
5.4.11 GetSDKVersion .....	65
5.4.12 GetDeviceIP .....	65
5.4.13 SetDeviceIP.....	66
5.4.14 GetDeviceMAC .....	66
5.4.15 SetDeviceMAC .....	67
5.4.16 GetWiegandFmt.....	67
5.4.17 SetWiegandFmt .....	68
5.4.18 GetCardFun.....	68
5.4.19 SetDeviceCommPwd .....	68
5.4.20 SetCommPassword .....	69
5.4.21 QueryState .....	69
5.4.22 GetVendor .....	70
5.4.23 GetDeviceStrInfo .....	70
5.4.24 GetPlatform .....	71
5.4.25 ReadAOptions .....	71
5.4.26 GetSysOption .....	72
5.4.27 SetSysOption.....	72
5.5 Others.....	73
5.5.1 Device Control Functions .....	73
5.5.2 Card Operation Functions .....	75
5.5.3 Others .....	76

## **6 FAQs .....81**

6.1 How to Download Attendance Records? .....	81
6.2 How to Create a User Online?.....	81
6.3 How to Import or Download Data from USB Disk? .....	81
6.4 How to Use Biokey to Write the Collected Fingerprint Templates Offline?.....	83
6.5 How to Obtain All Information of All Users? .....	83
6.6 How to Connect to the Device? .....	84
6.7 Password Is Invalid After SSR_SetUserInfo Is Used.....	84
6.8 How to Convert an Online Template into an Offline Template? .....	84
6.9 Demo Program Fails to Connect to the Device. ....	84
6.10 Standalone fingerprint Device Keeps Working After Being Connected. ....	85
6.11 Illegal Characters Are Displayed or Screen Display Is Abnormal After Non-English Names or Short Messages Are Uploaded to the Device.....	85
6.12 Card Management Problems.....	85
6.13 Firewall or Router Traversal .....	86
6.14 Uploading a Large Capacity of Fingerprints .....	86

6.15 Differences between High-speed Upload and Ordinary Upload.....	86
6.16 How to Download a Face Template?.....	86

# 1 SDK Description

The communication protocol SDK is an interface for data communication with standalone fingerprint devices, access control devices, and RFID card devices. It can be used to conveniently manage user information and fingerprints, download attendance records, operation records, user information, fingerprint templates, and face templates, set devices, and configure access control. The SDK is used to:

1. Download attendance records.
2. Upload and download user information, card information, fingerprints, and face information.
3. Set access control rules of access control devices.
4. Set device time, match thresholds, etc.
5. Trigger various events of devices in real time, for example, fingerprint verification.

## 2 Quick Start

### 2.1 Terms

1. Real-time event

After the SDK and the device communicate with each other successfully, some operations on the device (for example, connecting to the device, verifying a user, and enrolling a user) trigger corresponding events in real time, and data is transmitted to the PC (host computer). The triggered events are called real-time events. Users can monitor device states and user operations in real time through real-time events.

2. FP

Shortened form of "fingerprint".

3. Fingerprint algorithm

A fingerprint algorithm refers to the algorithm used to generate and verify fingerprint templates. At present, ZKFinger 10.0 is the latest fingerprint algorithm. It is a high-speed algorithm with higher performance. For details, see FAQs.

4. High-speed buffer

A high-speed buffer refers to the memory requested by the SDK on a PC during usage. In the data upload or download process, data is first saved in the buffer before being processed.

5. Time Zone, group, open door combination

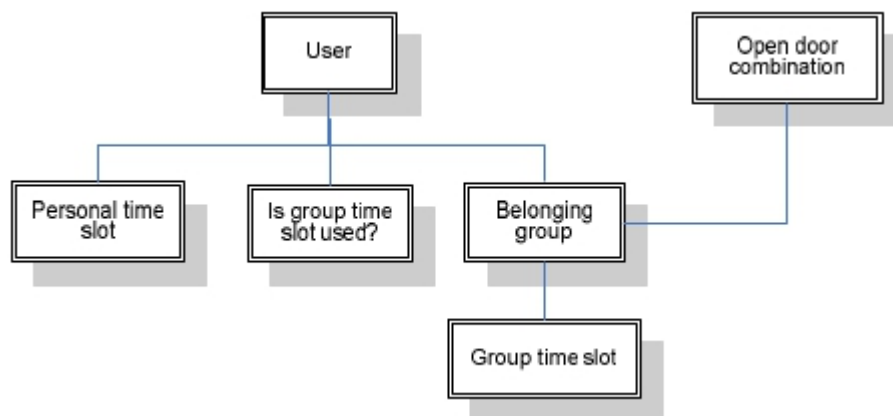
These three terms are the most important concepts of access control.

A Time Zone is a time range. A Time Zone includes the time information of one week, and a time range is specified for each day of this week. For example, the following expression indicates a time range from 00:00 to 22:11 in each day of one week: 000022110000221100002211000022110000221100002211. Generally, 50 Time Zones can be set in the device.

A group is a collection. When many users have the same access control privileges, these users can be added to the same group and use the group Time Zone. Then, Time Zones can be set for the group.

An open door combination refers to the groups that are required for unlock. If the open door combination contains only one group, it indicates that the door is opened when any of the users in this group passes verification. If the open door combination contains two or more groups, the door is opened only after all groups pass verification. For example, an open door combination contains groups A and B, the door is opened only after a member of group A and a member of group B pass verification.

The following figure shows the relationship of the three concepts:



6. Operation record

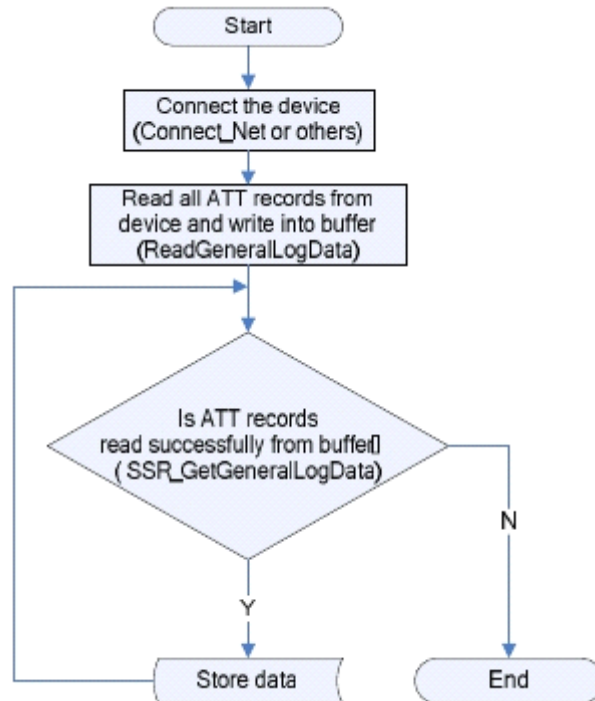
An operation record, also called management record, is a record generated when users or administrators operate on the device, for example, powering on/off the device and enrolling a user.



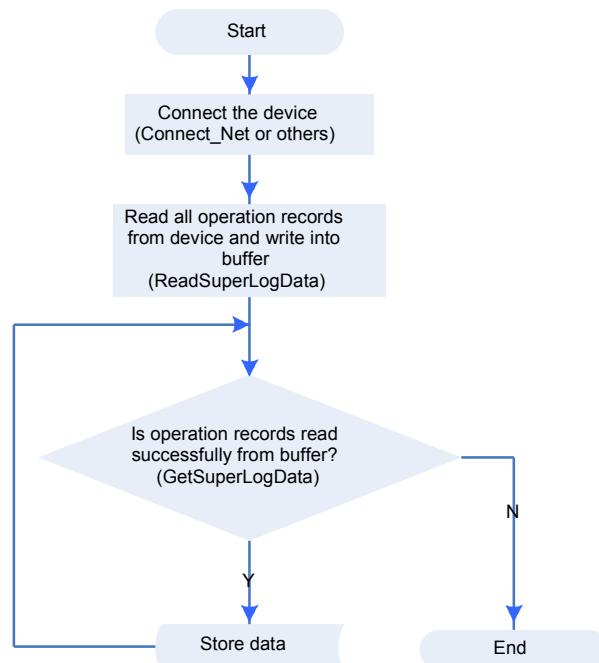
## 2.2 Common Processes

For details, see the descriptions of the demo program.

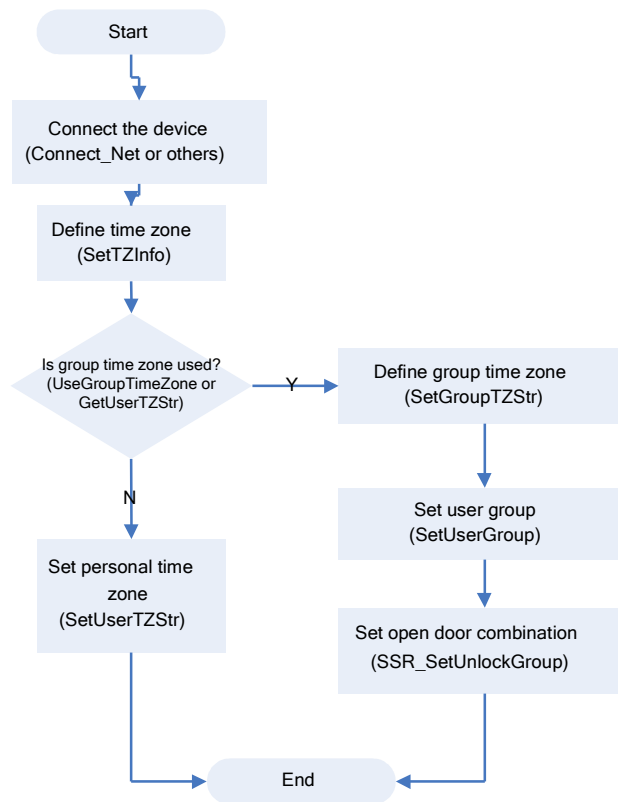
### 2.2.1 Downloading Attendance Records



### 2.2.2 Downloading Operation Records

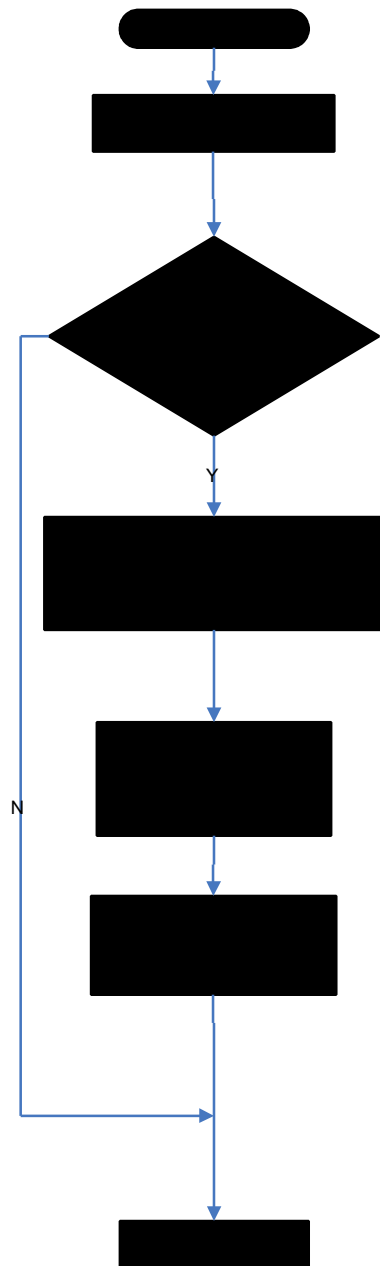
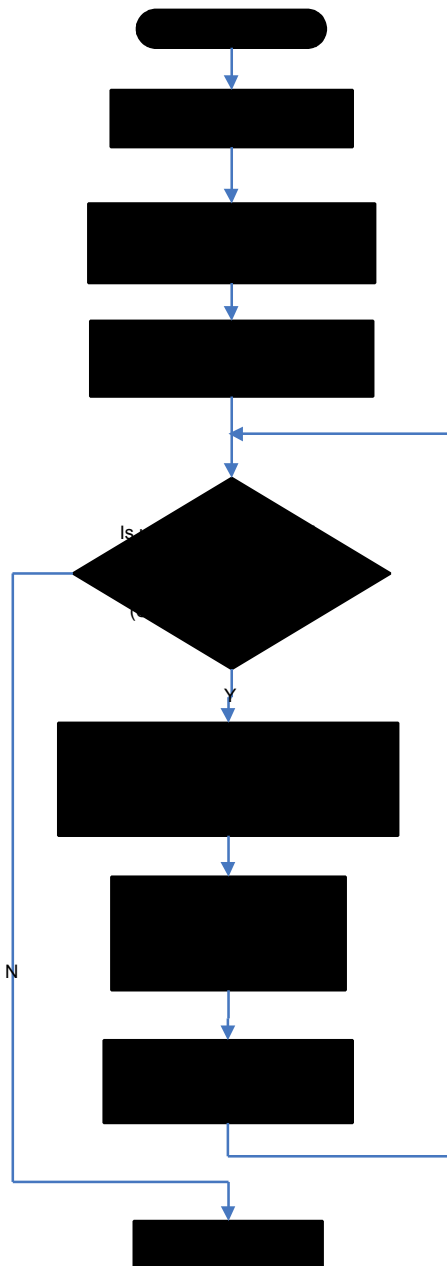


### 2.2.3 Setting Access Control



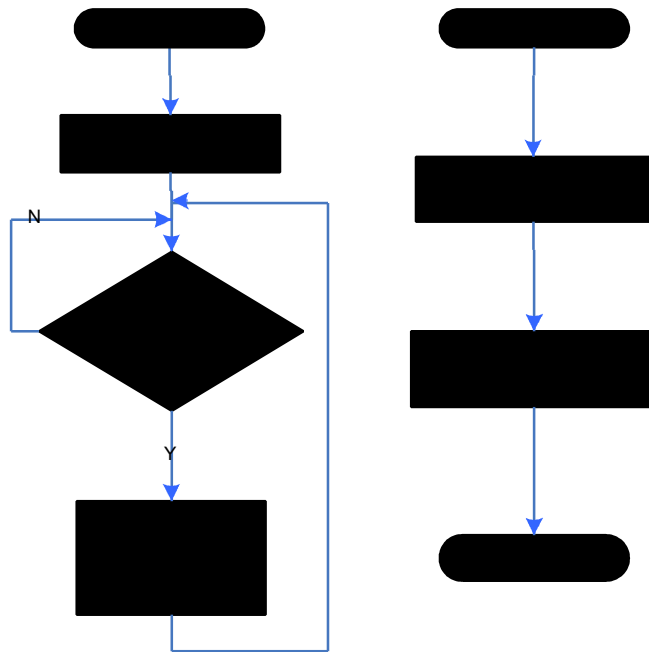
### 2.2.4 Downloading User Information, Fingerprint Templates, or Face Templates

The left diagram shows the process of downloading the information of all users. The right diagram shows the process of downloading the information of a specified user.



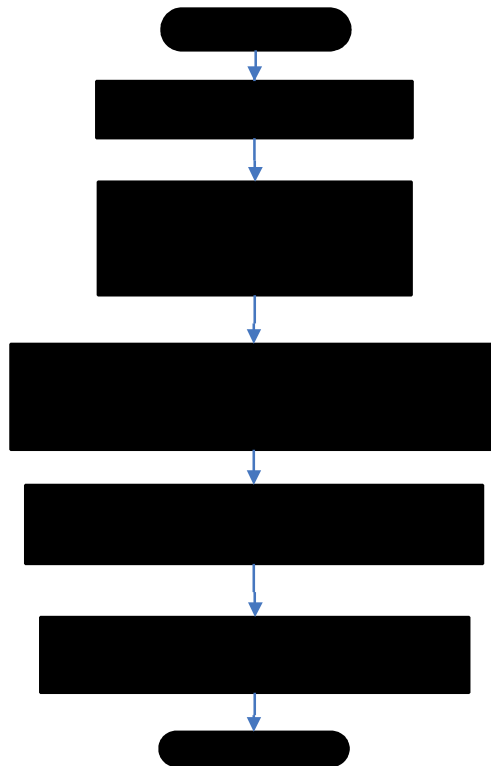
### 2.2.5 Receiving Real-time Events

Real-time events can be received in two modes. The second mode is recommended.



### 2.2.6 Enrolling Users Online (Uploading Information, Fingerprint Templates, and Face Templates of Users)

The following diagram shows the process of creating a user on the device and uploading the card number, password, and fingerprint information for the user (that is, enrolling a card user, a password user, and a fingerprint user).



## 3 Related Attributes

### 3.1 AccGroup

**Usage:** Set or obtain the group to which a user belongs.

If this attribute is set before user information is uploaded, the user group is set when SetUserInfo is used to upload user information. Otherwise, AccGroup is 1 by default.

**Type:** LONG, writable

### 3.2 AccTimeZones

**Usage:** Set the Time Zone of a user.

If this attribute is set before user information is uploaded, the Time Zone of a user is set when SetUserInfo is used to upload user information.

**Type:** LONG \*. It can be regarded as a long one-dimensional array with subscript 3.  
Readable/writable

### 3.3 BASE64

**Usage:** When the value is set to True, the SDK outputs the character string template in base64 codes. Otherwise, the SDK outputs the character string template in hexadecimal codes.

**Type:** LONG, readable/writable

### 3.4 CardNumber

**Usage:** Set or read the card number of a user. If this attribute is unavailable, use GetStrCardnumber and SetStrCardnumber instead.

**Type:** LONG, readable/writable

### 3.5 CommPort

**Usage:** Set the number of the serial port or the port for RS485 communication.

**Type:** LONG, readable/writable

### 3.6 ConvertBIG5

**Usage:** When the value is set to True, the SDK automatically converts the font from simplified Chinese to traditional Chinese for offline development. This function is invalid for series products of Multilanguage versions. Do not set this function in this case.

**Type:** LONG, readable/writable

**Caution:** This attribute is invalid for Multilanguage versions. In addition, you do not need to modify this attribute for ZEM100 5.22, ZEM200 5.30, and later versions.

### 3.7 PINWidth

**Usage:** Indicate the maximum length of the user ID (Arabic numeral).

**Type:** LONG, read only

### 3.8 GetStrCardNumber

#### [Definition]

VARIANT\_BOOL GetStrCardNumber([out] BSTR\* ACardNumber)

#### [Usage]

Obtain the value of cardnumber of the SDK. Generally, this function can be used to obtain the card number of a user immediately after the user information is obtained.

#### [Parameter]

AcardNumber

Card number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetUserInfo

## 3.9 SetStrCardNumber

**[Definition]**

VARIANT\_BOOL SetStrCardNumber([in] BSTR ACardNumber)

**[Usage]**

Set the value of cardnumber of the SDK. Generally, this function can be used to set the card number of a user before the user information is set.

**[Parameter]**

AcardNumber

Card number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetUserInfo

## 4 Real-time Event Functions

### 4.1 Obtaining Real-Time Events

#### 4.1.1 RegEvent

**[Definition]**

VARIANT\_BOOL RegEvent( [in] LONG dwMachineNumber, [in] LONG EventMask)

**[Usage]**

Register required real-time events.

**[Parameter]**

dwMachineNumber:

Device number

EventMask:

Code of an event. Values are as follows:

1	OnAttTransaction, OnAttTransactionEx
2 (1<<1)	OnFinger
4 (1<<2)	OnNewUser
8 (1<<3)	OnEnrollFinger
16 (1<<4)	OnKeyPress
256 (1<<7)	OnVerify
512 (1<<8)	OnFingerFeature
1024 (1<<9)	OnDoor, OnAlarm
2048 (1<<10)	OnHIDNum
4096 (1<<11)	OnWriteCard
8192 (1<<12)	OnEmptyCard
16384 (1<<13)	OnDeleteTemplate

To register multiple real-time events, perform the XOR operation between binary codes of related events. For example, to register all real-time events, the value of EventMask is 65535.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ReadRTLog, GetRTLog

## 4.1.2 ReadRTLog

**[Definition]**

VARIANT\_BOOL ReadRTLog( [in] LONG dwMachineNumber)

**[Usage]**

Read real-time events and write them to the buffer of the PC. This function can be used with GetRTLog to actively obtain real-time events from the device after the PC connects to the device successfully.

**[Parameter]**

dwMachineNumber:  
Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetRTLog

## 4.1.3 GetRTLog

**[Definition]**



VARIANT\_BOOL GetRTLog(LONG dwMachineNumber)

**[Usage]**

Obtain an event from the buffer of the PC and trigger the event. This function can be used with ReadRTLog to actively obtain real-time events from the device after the PC connects to the device successfully.

**[Parameter]**

dwMachineNumber:  
Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ReadRTLog

## 4.2 Real-Time Events

### 4.2.1 OnConnected

This event is triggered when the PC connects to the device successfully. No value is returned.

### 4.2.2 OnDisconnected

This event is triggered when the PC disconnects from the device successfully. No value is returned.

### 4.2.3 OnAlarm

OnAlarm (LONG AlarmType, LONG EnrollNumber, LONG Verified)

This event is triggered when the device reports an alarm.

**[Return Value]**

Alarm Type: Type of an alarm. 55: Tamper alarm. 58: **False alarm**. 32: **Threatened alarm**. 34: Anti-pass back alarm.

EnrollNumber: User ID. The value is 0 when a tamper alarm, false alarm, or threatened alarm is given. The value is the user ID when other threatened alarm or anti-pass back alarm is given.

Verified: Whether to verify The value is 0 when a tamper alarm, **false alarm**, or threatened alarm is given. The value is 1 when other alarms are given.

### 4.2.4 OnDoor

OnDoor (LONG EventType)

This event is triggered when the device opens the door.

### **[Return Value]**

EventType: Open door type

4: The door is not closed. 53: Exit button. 5: The door is closed. 1: The door is opened unexpectedly.

## **4.2.5 OnAttTransactionEx**

OnAttTransactionEx (BSTR EnrollNumber, LONG IsInvalid, LONG AttState, LONG VerifyMethod, LONG Year, LONG Month, LONG Day, LONG Hour, LONG Minute, LONG Second, LONG WorkCode)

This event is triggered after verification succeeds.

### **[Return Value]**

EnrollNumber: UserID of a user.

IsInvalid: Whether a record is valid. 1: Not valid. 0: Valid.

AttState: Attendance state (default value).      0—Check-In    1—Check-Out    2—Break-Out  
3—Break-In 4—OT-In 5—OT-Out

VerifyMethod: Verification mode. Generally,      0: password verification,    1: fingerprint verification, 2: card verification.

In multi-verification mode:

0(FP/PW/RF), 1(FP), 2(PIN), 3(PW), 4(RF), 5(FP&RF),  
6(FP/PW), 7(FP/RF), 8(PW/RF), 9(PIN&FP), 10(FP&PW), 11(PW&RF),  
12(FP&PW&RF), 13(PIN&FP&PW), 14(FP&RF/PIN), 15(FACE), 16(FACE&FP), 17(FACE&PW),  
18(FACE&RF), 19(FACE&FP&RF), 20(FACE&FP&PW)

Year/Month/Day/Hour/Minute/ Second indicates the time when verification succeeds.

WorkCode: work code returned during verification. Return 0 when the device does not support work code.

#### 4.2.6 OnEnrollFingerEx

OnEnrollFinger (LONG EnrollNumber, LONG FingerIndex, LONG ActionResult, LONG TemplateLength)

This event is triggered when a fingerprint is registered.

**[Return Value]**

EnrollNumber: User ID of the fingerprint being registered

FingerIndex: Index of the current fingerprint

ActionResult: Operation result. Return 0 if the operation is successful, or return a value greater than 0.

TemplateLength: Length of the fingerprint template

#### 4.2.7 OnFinger

This event is triggered when a fingerprint is placed on the fingerprint sensor of the device. No value is returned.

#### 4.2.8 OnFingerFeature

OnFingerFeature (LONG Score)

This event is triggered when a user places a finger and the device registers the fingerprint.

**[Return Value]**

Score: Quality score of a fingerprint

#### 4.2.9 OnHIDNum

OnHIDNum (LONG CardNumber)

This event is triggered when a card is swiped.

**[Return Value]**

CardNumber: Number of a card that can be an ID card or an HID card. If the card is a Mifare card, the event is triggered only when the card is used as an ID card.

#### 4.2.10 OnNewUser

OnNewUser (LONG EnrollNumber)

This event is triggered when a new user is successfully enrolled.

**[Return Value]**

EnrollNumber: UserID of the newly enrolled user.

#### 4.2.11 OnVerify

OnVerify (LONG UserID)

This event is triggered when a user is verified.

**[Return Value]**

When verification succeeds, UserID indicates the ID of the user. Return the card number if card verification is successful, or return -1.

**4.2.12 OnWriteCard**

OnWriteCard (LONG EnrollNumber, LONG ActionResult, LONG Length)

This event is triggered when the device writes a card.

**[Return Value]**

EnrollNumber: ID of the user to be written into a card

ActionResult: Result of writing user information into a card. 0: Success. Other values: Failure.

Length: Size of the data to be written into a card

**4.2.13 OnEmptyCard**

OnEmptyCard (LONG ActionResult)

This event is triggered when a Mifare card is emptied.

**[Return Value]**

ActionResult: Result of emptying a card. 0: Success. Other values: Failure.

**4.2.14 OnEMData**

OnEMData (LONG DataType, LONG DataLen, CHAR\* DataBuffer)

This event is triggered when the device sends an unknown event to SDK.

**[Return Value]**

DataType: Type of the returned event

DataLen: Data length

DataBuffer: Actual data

## **5 Common Functions**

### **5.1 Device Connection Functions**

#### **5.1.1 Connect\_Net**

**[Definition]**

VARIANT\_BOOL Connect\_Net( [in] BSTR IPAdd, [in] long PortI)

**[Usage]**

Connect to the device via the IP address and set up a network connection with the device.

**[Parameter]**

IPAdd:

IP address of the device

Port:

Port number used for connecting to the device. The default value is 4370.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

Disconnect, Connect\_Com, Connect\_USB

### 5.1.2 Connect\_Com

**[Definition]**

VARIANT\_BOOL Connect\_Com( [in] long ComPort, [in] long MachineNumber, [in] long BaudRate)

**[Usage]**

Connect to the device via a serial port, that is, via RS232 or RS485 port.

**[Parameter]**

ComPort:

Serial port number of the PC for connecting to the device

MachineNumber:

Device number

BaudRate:

Baud rate

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

Disconnect, Connect\_Net

### 5.1.3 Disconnect

**[Definition]**

Disconnect(void)

**[Usage]**

Disconnect from the device and release related resources.

**[Parameter]**

None

**[Return Value]**

None

**[Related Function]**

Connect\_Net, Connect\_Com

## 5.2 Data Management Functions

### 5.2.1 Attendance Record Data

#### 5.2.1.1 ReadGeneralLogData

**[Definition]**

VARIANT\_BOOL ReadGeneralLogData( [in] long dwMachineNumber)

**[Usage]**

Read attendance records and write them into the internal buffer of the PC. This function is the same as ReadAllGLogData.

**[Parameter]**

dwMachineNumber:

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ReadAllGLogData, SSR\_GetGenLogData, ClearGLog

#### 5.2.1.2 ReadAllGLogData

**[Definition]**

VARIANT\_BOOL ReadAllGLogData ( [in] long dwMachineNumber)

**[Usage]**

Read attendance records and write them into the internal buffer of the PC. This function is the same as ReadGeneralLogData.

**[Parameter]**

dwMachineNumber:

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ReadGeneralLogData、SSR\_GetGeneralLogData、ClearGLog

#### 5.2.1.3 SSR\_GetGeneralLogData

**[Definition]**

VARIANT\_BOOL SSR\_GetGeneralLogData( [in] LONG dwMachineNumber, [out] BSTR\* dwEnrollNumber, [out] LONG\* dwVerifyMode, [out] LONG\* dwInOutMode, [out] LONG\* dwYear, [out] LONG\* dwMonth, [out] LONG\* dwDay, [out] LONG\* dwHour, [out] LONG\* dwMinute, [out] LONG\* dwSecond, [out] LONG\* dwWorkcode)

### **[Usage]**

Read attendance records one by one from the internal buffer. Before using this function, you can use ReadAllGLogData or ReadGeneralLogData to read attendance records from the device and write them into the internal buffer of the PC. Each time this function is executed, the pointer points to the next attendance record.

### **[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

Pointer pointing to a BSTR variable. The value is the user ID of the received attendance record. The value consists of up to 24 bits.

dwVerifyMode

Pointer pointing to a long variable. The value is the verification mode of the received attendance record. The values are as follows:

Generally, 0: password verification, 1: fingerprint verification, 2: card verification.

In multi-verification mode:

0(FP/PW/RF), 1(FP), 2(PIN), 3(PW), 4(RF), 5(FP&RF),

6(FP/PW), 7(FP/RF), 8(PW/RF), 9(PIN&FP), 10(FP&PW), 11(PW&RF),

12(FP&PW&RF), 13(PIN&FP&PW), 14(FP&RF/PIN), 15(FACE), 16(FACE&FP), 17(FACE&PW),

18(FACE&RF), 19(FACE&FP&RF), 20(FACE&FP&PW)

Pointer pointing to a long variable. The value is the AttState of the received attendance record. The values are as follows:

0—Check-In (default value)    1—Check-Out 2—Break-Out  
3—Break-In    4—OT-In    5—OT-Out

dwYear, dwMonth, dwDay, dwHour, dwMinute, dwSecond

Pointers pointing to long variables. The values are the date and time of the received attendance record.

dwWorkcode

Pointer pointing to a long variable. The value is the work code of the received attendance record.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ReadGeneralLogData

#### **5.2.1.4 ClearGLog**

**[Definition]**

VARIANT\_BOOL ClearGLog([in] long dwMachineNumber)

**[Usage]**

Clear all attendance records from the device.

**[Parameter]**

dwMachineNumber

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ClearSLog, ClearKeeperData

### **5.2.2 Operation Record Data**

#### **5.2.2.1 ReadSuperLogData**

**[Definition]**

VARIANT\_BOOL ReadSuperLogData( [in]long dwMachineNumber)

**[Usage]**

Read operation records and write them into the internal buffer of the PC. This function is the same as ReadAllSLogData.

**[Parameter]**

dwMachineNumber:



Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ReadAllSLogData, GetAllSLogData, GetSuperLogData, ClearSLog

### 5.2.2.2 ReadAllSLogData

**[Definition]**

VARIANT\_BOOL ReadAllSLogData( [in] long dwMachineNumber)

**[Usage]**

Read operation records and write them into the internal buffer of the PC. This function is the same as ReadSuperLogData.

**[Parameter]**

dwMachineNumber:

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ReadSuperLogData, GetAllSLogData, GetSuperLogData, ClearSLog

### 5.2.2.3 GetSuperLogData

**[Definition]**

VARIANT\_BOOL GetSuperLogData( [in] long dwMachineNumber, [out] long\* dwTMachineNumber, [out] long\* dwSEnrollNumber, [out] long\* Params4, [out] long\* Params1, [out] long\* Params2, [out] long\* dwManipulation, [out] long\* Params3, [out] long\* dwYear, [out] long\* dwMonth, [out] long\* dwDay, [out] long\* dwHour, [out] long\* dwMinute)

**[Usage]**

Read operation records one by one from the internal buffer. Before using this function, you can use ReadAllSLogData or ReadSuperLogData to read operation records from the device and write them into the internal buffer of the PC. Each time this function is executed, the pointer points to the next operation record. This function is the similar to GetSuperLogData2 except that GetSuperLogData2 can be used to obtain more accurate operation record time (in seconds).

**[Parameter]**

dwMachineNumber

Device number

dwTMachineNumber

Pointer pointing to a long variable. The value is the device number of the received operation record.

dwSEnrollNumber

Pointer pointing to a long variable. The value is the administrator ID of the received operation record.

Params4

Pointer pointing to a long variable. The value varies with dwManipulation.

Params1

Pointer pointing to a long variable. The value varies with dwManipulation.

Params2

Pointer pointing to a long variable. The value varies with dwManipulation.

dwManipulation

Pointer pointing to a long variable. The value is the operation type. The specific values are as follows:

Value of dwManipulation	Meaning of dwManipulation	Params1	Params2	Params3	Params4
0	Power on				
1	Power off				
3	Alarm	Alarm type. 58: False alarm. 54: Door entry alarm, 53: Exit button alarm, 55: Tamper alarm, 65535: Alarm off			
4	Enter menu				
5	Change setting	Number of the set option			
6	Enroll a fingerprint	ID of the operated user	Operation result. 0: Success. Other values: Failure.	Index of the registered fingerprint	Length of the fingerprint template (2: Threatened fingerprint)
7	Enroll a password				

Value of dwManipulation	Meaning of dwManipulation	Params1	Params2	Params3	Params4
14	Create an MF card	ID of the operated user		Number of fingerprints written into the MF card	Size of fingerprint data written into the MF card
20	Copy data from the MF card to the device			Number of fingerprints read out from the MF card	
22	Restore factory setting				
30	Enroll a new user	ID of the operated user	Operation result		
32	Threatened alarm	Whether to verify. 0: Key alarm. 1: Verify alarm	Note: When the value is Verify alarm, dwSEnrollNumber returns the ID of the threatened user.		
34	Anti-pass back	Whether the alarm is an anti-pass back alarm.			

Params3

Pointer pointing to a long variable. The value varies with dwManipulation.

dwYear, dwMonth, dwDay, dwHour, dwMinute

Pointers pointing to long variables. The values are the date and time of the received operation record.

#### [Return Value]

Return True if it is successful, or return False.

#### [Related Function]

GetSuperLogData2, GetAllSLogData

### 5.2.2.4 GetAllSLogData

#### [Definition]

GetAllSLogData( [in] long dwMachineNumber, [out] long\* dwTMachineNumber, [out] long\* dwSEnrollNumber, [out] long\* dwSMachineNumber, [out] long\* dwGEnrollNumber, [out] long\* dwGMachineNumber, [out] long\* dwManipulation, [out] long\* dwBackupNumber, [out]

long\* dwYear, [out] long\* dwMonth, [out] long\* dwDay, [out] long\* dwHour, [out] long\* dwMinute)

**[Usage]**

Read operation records one by one from the internal buffer. Before using this function, you can use ReadAllSLogData or ReadSuperLogData to read operation records from the device and write them into the internal buffer of the PC. Each time this function is executed, the pointer points to the next operation record. This function is the same as GetSuperLogData except that interface names are different to achieve compatibility.

**[Parameter]**

Same as GetSuperLogData

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetSuperLogData

### 5.2.2.5 ClearSLog

**[Definition]**

VARIANT\_BOOL ClearSLog([in] long dwMachineNumber)

**[Usage]**

Clear all operation records from the device.

**[Parameter]**

dwMachineNumber  
Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ClearGLog, ClearKeeperData

### 5.2.2.6 GetSuperLogData2

**[Definition]**

VARIANT\_BOOL GetSuperLogData2([in] LONG dwMachineNumber, [out] LONG\* dwTMachineNumber, [out] LONG\* dwSEnrollNumber, [out] LONG\* Params4, [out] LONG\* Params1, [out] LONG\* Params2, [out] LONG\* dwManipulation, [out] LONG\* Params3, [out] LONG\* dwYear, [out] LONG\* dwMonth, [out] LONG\* dwDay, [out] LONG\* dwHour, [out] LONG\* dwMinute, [out] LONG\* dwSecs)

**[Usage]**

Read operation records one by one from the internal buffer. Before using this function, you can use ReadAllSLogData or ReadSuperLogData to read operation records from the device and write them into the internal buffer of the PC. Each time this function is executed, the

pointer points to the next operation record. This function is the similar to GetSuperLogData except that GetSuperLogData can be used to obtain more accurate operation record time (in seconds).

**[Parameter]**

dwYear, dwMonth, dwDay, dwHour, dwMinute, dwSecs

Pointers pointing to long variables. The values are the date and time of the received operation record.

For other parameters, see GetSuperLogData.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ReadAllSLogData, GetSuperLogData

## 5.2.3 User Information Functions

### 5.2.3.1 ReadAllUserID

**[Definition]**

VARIANT\_BOOL ReadAllUserID([in] long dwMachineNumber)

**[Usage]**

Read all user information except fingerprint templates, such as user ID, password, name, and card number, and write the user information into the memory of the PC. After this function is executed, you can use GetAllUserID to take out user information.

**[Parameter]**

dwMachineNumber

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetAllUserID

### 5.2.3.2 SSR\_EnableUser

**[Definition]**

VARIANT\_BOOL SSR\_EnableUser([in] LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in] VARIANT\_BOOL bFlag)

**[Usage]**

Set whether a user is enabled.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

bFlag

User enable flag. True: Enabled. False: Disabled.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

**5.2.3.3 SetUserInfoEx**

**[Definition]**

VARIANT\_BOOL SetUserInfoEx([in] LONG dwMachineNumber, [in] LONG dwEnrollNumber, [in] LONG VerifyStyle, [in] BYTE\* Reserved)

**[Usage]**

Upload the user verification mode or group verification mode. Only the devices supporting multiple verification modes support this function.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

VerifyStyle

Verification mode

For TFT access control fingerprint devices: 0 (group verification mode), 128(FP/PW/RF), 129(FP), 130(PIN), 131(PW), 132(RF), 133(FP&RF), 134(FP/PW), 135(FP/RF), 136(PW/RF), 137(PIN&FP), 138(FP&PW), 139(PW&RF), 140(FP&PW&RF), 141(PIN&FP&PW), 142(FP&RF/PIN), 143(FACE), 144(FACE&FP), 145(FACE&PW), 146(FACE&RF), 147(FACE&FP&RF), 148(FACE&FP&PW).

Reserved

Reserved parameter temporarily without specific meanings.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetUserInfoEx, DeleteUserInfoEx

**5.2.3.4 GetUserInfoEx**

**[Definition]**

VARIANT\_BOOL      GetUserInfoEx([in]    LONG    dwMachineNumber,    [in]    LONG  
dwEnrollNumber, [out] LONG\* VerifyStyle, [out] BYTE\* Reserved)

**[Usage]**

Obtain the user verification mode. Only the devices support multiple verification modes support this function.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

VerifyStyle

Verification mode of the user described by dwEnrollNumber. The values are as follows:

For TFT access control fingerprint device: 0 (group verification mode), 128(FP/PW/RF), 129(FP), 130(PIN), 131(PW), 132(RF), 133(FP&RF), 134(FP/PW), 135(FP/RF), 136(PW/RF), 137(PIN&FP), 138(FP&PW), 139(PW&RF), 140(FP&PW&RF), 141(PIN&FP&PW), 142(FP&RF/PIN), 143(FACE), 144(FACE&FP), 145(FACE&PW), 146(FACE&RF), 147(FACE&FP&RF), 148(FACE&FP&PW).

Reserved

Reserved parameter without specific meanings.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetUserInfoEx, DeleteUserInfoEx

**5.2.3.5 DeleteUserInfoEx**

**[Definition]**

VARIANT\_BOOL      DeleteUserInfoEx([in]    LONG    dwMachineNumber,    [in]    LONG  
dwEnrollNumber)

**[Usage]**

Delete the multiple verification modes of the specified user. Only the devices supporting multiple verification modes support this function.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetUserInfoEx, GetUserInfoEx

**5.2.3.6 SSR\_GetAllUserInfo****[Definition]**

```
VARIANT_BOOL      GetAllUserInfo([out] long dwMachineNumber, [out] long*  
dwEnrollNumber, [out] BSTR * Name, [out] BSTR * Password, [out] long * Privilege, [out]  
VARIANT_BOOL * Enabled)
```

**[Usage]**

Obtain all user information. Before executing this function, you can use ReadAllUserID to read out all user information and write it into the memory. Each time SSR\_GetAllUserInfo is executed, the pointer points to the next user information. After all user information is read, False is returned.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

Name

User name

Password

User password

Privilege

User privilege. 3: super administrator 0: common user,

dwEnable

Whether the user is enabled. 1: Enabled. 0: Disabled.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ReadAllUserID, GetAllUserID

**5.2.3.7 SSR\_GetUserInfo****[Definition]**

```
VARIANT_BOOL      SSR_GetUserInfo([in] LONG dwMachineNumber, [in] BSTR  
dwEnrollNumber, [out] BSTR* Name, [out] BSTR* Password, [out] LONG* Privilege, [out]  
VARIANT_BOOL* Enabled)
```

**[Usage]**

Obtain the information of the specified user.



**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

Name

Name of the user described by dwEnrollNumber

Password

Password of the user described by dwEnrollNumber

Privilege

Privilege of the user described by dwEnrollNumber. 3: administrator, 0: common user

Enabled

User enable flag. 1: Enabled. 0: Disabled

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SSR\_SetUserInfo

**5.2.3.8 SSR\_SetUserInfo****[Definition]**

VARIANT\_BOOL SetUserInfo([in] long dwMachineNumber, [in] long dwEnrollNumber, [in] BSTR Name, [in] BSTR Password, [in] long Privilege, [in] VARIANT\_BOOL Enabled)

**[Usage]**

Set user information. If the user is unavailable, the device automatically creates the user.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

Name

User name

Password

User password

Privilege

User privilege. 3: super administrator, 0: common user

Enabled

User enable flag. 1: Enabled. 0: Disabled

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetUserInfo

## **5.2.4 Enroll Data Functions (Including Both User Information and Fingerprint)**

### **5.2.4.1 SSR\_DeleteEnrollData**

**[Definition]**

VARIANT\_BOOL DeleteEnrollData([in] long dwMachineNumber, [in] long dwEnrollNumber, [in] long dwEMachineNumber, [in] long dwBackupNumber)

**[Usage]**

Delete enrollment data.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

dwBackupNumber

Index of the fingerprint

The value ranges from 0 to 9. In this case, the device also checks whether the user has other fingerprints or passwords. If no, the device deletes the user.

When the value is 10, the device deletes the password. The device also checks whether the user has fingerprint data. If no, the device deletes the user.

When the value is 11, the device deletes all fingerprint data of the user. When the value is 12, the device deletes the user (including all fingerprints, card numbers and passwords of the user).

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SSR\_SetUserInfo、SSR\_SetUserTmp、SSR\_DeleteEnrollDataExt

### **5.2.4.2 SSR\_DeleteEnrollDataExt**

**[Definition]**

VARIANT\_BOOL SSR\_DeleteEnrollDataExt([in] LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in] LONG dwBackupNumber)

**[Usage]**

Delete enrollment data. The difference between this function and SSR\_DeleteEnrollData is that this function can delete all fingerprints. This function achieves a higher efficiency.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

dwBackupNumber

Index of the fingerprint

The value ranges from 0 to 9. In this case, the device also checks whether the user has other fingerprints or passwords. If no, the device deletes the user.

When the value is 0, the device deletes the password. The device also checks whether the user has fingerprint data. If no, the device deletes the user.

When the value is 11 or 13, the device deletes all fingerprint data of the user. When the value is 12, the device deletes the user (including all fingerprints, card numbers and passwords of the user).

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SSR\_SetUserInfo, SSR\_SetUserTmp, SSR\_DeleteEnrollData

## 5.2.5 Fingerprint Template Functions

### 5.2.5.1 ReadAllTemplate

**[Definition]**

VARIANT\_BOOL ReadAllTemplate([in] LONG dwMachineNumber)

**[Usage]**

Read out all fingerprint templates from the device and write them into the memory of the PC. This function is used to read out and write all the fingerprints into the memory at a time. This function achieves a higher speed in comparison with the way of obtaining fingerprints one by one.

**[Parameter]**

dwMachineNumber

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

**5.2.5.2 SSR\_DelUserTmp****[Definition]**

VARIANT\_BOOL DelUserTmp([in] long dwMachineNumber, [in] long dwEnrollNumber, [in] long dwFingerIndex)

**[Usage]**

Delete the fingerprint template of a specified user.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

dwFingerIndex

Index of the fingerprint. The value ranges from 0 to 9.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SSR\_SetUserTmp、SSR\_GetUserTmp

**5.2.5.3 SSR\_DelUserTmpExt****[Definition]**

VARIANT\_BOOL SSR\_DelUserTmpExt([in] LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in] LONG dwFingerIndex)

**[Usage]**

Delete the fingerprints of a specified user. The difference between this function and DelUserTmp is that DelUserTmp supports 24-bit user ID.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

dwFingerIndex

Index of the fingerprint. The value ranges from 0 to 9.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SSR\_DelUserTmp

**5.2.5.4 SetUserTmpEx****[Definition]**

VARIANT\_BOOL SetUserTmpEx([in] LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in] LONG dwFingerIndex, [in] LONG Flag, [in] BYTE\* TmpData)

**[Usage]**

Upload ordinary fingerprint templates or threatened fingerprint templates of a user in binary mode. The only difference between this function and SetUserTmpExStr is the fingerprint template format. Caution: The user must have been created on the device, or the user information must be uploaded together with the fingerprint templates. If the template with the same index is already registered by the same user, the fingerprint template will be overwritten. Note: TFT devices supporting threatened fingerprints (with firmware version 6.60 or later) can support this function.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

dwFingerIndex

Fingerprint index

Flag

Flag used to indicate whether the fingerprint template is valid or is a duress fingerprint template. 0: Invalid; 1: Valid; 3: duress fingerprint template.

TmpData

Fingerprint template data

TmpLength

Fingerprint template length

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetUserTmpEx, SSR\_SetUserInfo, BeginBatchUpdate, BatchUpdate

#### 5.2.5.5 SetUserTmpExStr

##### [Definition]

VARIANT\_BOOL SetUserTmpExStr([in] LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in] LONG dwFingerIndex, [in] LONG Flag, [in] BSTR TmpData)

##### [Usage]

Upload ordinary fingerprint templates or threatened fingerprint templates of a user in string mode. The only difference between this function and SetUserTmpEx is the fingerprint template format. Caution: The user must have been created on the device, or the user information must be uploaded together with the fingerprint templates. If the template with the same index is already registered by the same user, the fingerprint template will be overwritten. Note: TFT devices supporting threatened fingerprints (with firmware version 6.60 or later) can support this function.

##### [Parameter]

dwMachineNumber

Device number

dwEnrollNumber

User ID

dwFingerIndex

Fingerprint index

Flag

Flag used to indicate whether the fingerprint template is valid or is a duress fingerprint template. 0: Invalid; 1: Valid; 3: duress fingerprint template.

TmpData

Fingerprint template data

TmpLength

Fingerprint template length

##### [Return Value]

Return True if it is successful, or return False.

##### [Related Function]

GetUserTmpExStr, SSR\_SetUserInfo, BeginBatchUpdate, BatchUpdate

#### 5.2.5.6 GetUserTmpEx

##### [Definition]

VARIANT\_BOOL GetUserTmpEx([in] LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in] LONG dwFingerIndex,[out] LONG \* Flag, [out] BYTE\* TmpData, [out] LONG\* TmpLength)

#### **[Usage]**

Download ordinary fingerprint templates or threatened fingerprint templates of a user in binary mode. The only difference between this function and GetUserTmpExStr is the fingerprint template format. Note: TFT devices supporting threatened fingerprints (with firmware version 6.60 or later) can support this function.

#### **[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

dwFingerIndex

Fingerprint index

Flag

Flag used to indicate whether the fingerprint template is valid or is a duress fingerprint template. 0: Invalid; 1: Valid; 3: duress fingerprint template.

TmpData

Fingerprint template data

TmpLength

Fingerprint template length

#### **[Return Value]**

Return True if it is successful, or return False.

#### **[Related Function]**

SetUserTmpEx, SSR\_GetUserInfo

### **5.2.5.7 GetUserTmpExStr**

#### **[Definition]**

VARIANT\_BOOL GetUserTmpExStr([in] LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in] LONG dwFingerIndex,[out] LONG \* Flag, [out] BSTR\* TmpData, [out] LONG\* TmpLength)

#### **[Usage]**

Download ordinary fingerprint templates or threatened fingerprint templates of a user in string mode. The only difference between this function and GetUserTmpEx is the fingerprint template format. Note: TFT devices supporting threatened fingerprints (with firmware version 6.60 or later) can support this function.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

dwFingerIndex

Fingerprint index

Flag

Flag used to indicate whether the fingerprint template is valid or is a duress fingerprint template. 0: Invalid; 1: Valid; 3: duress fingerprint template.

TmpData

Fingerprint template data

TmpLength

Fingerprint template length

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetUserTmpExStr, SSR\_GetUserInfo

## 5.2.6 Face Data Functions

### 5.2.6.1 SetUserFace

**[Definition]**

VARIANT\_BOOL SetUserFace([in]LONG dwMachineNumber,[in]BSTR dwEnrollNumber,[in]LONG dwFaceIndex, [in] BYTE\* TmpData,[in] LONG TmpLength)

**[Usage]**

Upload face templates of a user. The only difference between this function is SetUserFaceStr is the face template format.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber



User ID (with less than 24 bits)

dwFaceIndex

Face index. The value can be only 50, that is, all face templates of the user are uploaded.

TmpData

Face template

TmpLength

Face template length

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetUserFace, DelUserFace, SetUserFaceStr, GetUserFaceStr

**5.2.6.2 GetUserFace**

**[Definition]**

VARIANT\_BOOL GetUserFace([in]LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in] LONG dwFaceIndex, [out] BYTE\* TmpData, [out] LONG \* TmpLength)

**[Usage]**

Download face templates. The only difference between this function and GetUserFaceStr is the format of the returned face template.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID (with less than 24 bits)

dwFaceIndex

Face index. The value can be only 50, that is, deleting all face templates of the user.

TmpData

Face template

TmpLength

Face template length

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetUserFace, DelUserFace, SetUserFaceStr, GetUserFaceStr

### 5.2.6.3 DelUserFace

#### [Definition]

VARIANT\_BOOL DelUserFace([in] LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in] LONG dwFaceIndex)

#### [Usage]

Delete the face templates of a user.

#### [Parameter]

dwMachineNumber

Device number

dwEnrollNumber

User ID (with less than 24 bits)

dwFaceIndex

Face index. The value can be only 50, that is, deleting all face templates of the user.

#### [Return Value]

Return True if it is successful, or return False.

#### [Related Function]

GetUserFace, SetUserFace, SetUserFaceStr, GetUserFaceStr

### 5.2.6.4 GetUserFaceStr

#### [Definition]

VARIANT\_BOOL GetUserFaceStr([in]LONG dwMachineNumber,[in]BSTR dwEnrollNumber, [in]LONG dwFaceIndex, [out]BSTR\* TmpData, [out]LONG \* TmpLength)

#### [Usage]

Download face templates. Difference from GetUserFace, this function is used to return face templates in string format.

#### [Parameter]

dwMachineNumber

Device number

dwEnrollNumber

User ID (with less than 24 bits)

dwFaceIndex

Face index. The value can be only 50, that is, downloading all face templates of the user.

TmpData

Face template

TmpLength

Face template length

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetUserFace, SetUserFace, SetUserFaceStr,

**5.2.6.5 SetUserFaceStr****[Definition]**

VARIANT\_BOOL SetUserFaceStr([in] LONG dwMachineNumber, [in] BSTR dwEnrollNumber, [in] LONG dwFaceIndex, [in] BSTR TmpData, [in] LONG TmpLength)

**[Usage]**

Upload face templates of a user. The only difference between this function is SetUserFace is the face template format.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID (with less than 24 bits)

dwFaceIndex

Face index. The value can be only 50, that is, uploading all face templates of the user.

TmpData

Face template

TmpLength

Face template length

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetUserFace, GetUserFaceStr, GetUserFace

**5.2.7 Fingerprint Template Conversion Functions****5.2.7.1 GetFPTempLength****[Definition]**

GetFPTempLength([in] BYTE\* dwEnrollData, [out] long \* Len)

**[Usage]**

Calculate the length of the specified fingerprint template.

**[Parameter]**

dwEnrollData

Pointer pointing to the fingerprint template

Len

Length of the fingerprint template described by dwEnrollData

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetFPTempLengthStr

**5.2.7.2 GetFPTempLengthStr**

**[Definition]**

GetFPTempLengthStr([in] BSTR dwEnrollData, [out] long \* Len)

**[Usage]**

Calculate the length of a specified fingerprint template.

**[Parameter]**

dwEnrollData

Fingerprint template in string form

Len

Length of the fingerprint template described by dwEnrollData

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetFPTempLength

**5.2.7.3 FPTempConvert**

**[Definition]**

VARIANT\_BOOL FPTempConvert([in] BYTE\* TmpData1, [out] BYTE\* TmpData2, [out] long \*Size)

**[Usage]**

Convert an Standalone fingerprint template into a Biokey fingerprint template. The only difference between this function and FPTempConvertStr is the data format.

**[Parameter]**

TmpData1

Standalone fingerprint template to be converted

TmpData2

Converted Biokey fingerprint template

Size

Size of the converted Biokey fingerprint template

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

FPTempConvertStr, FPTempConvertNew, FPTempConvertNewStr

**5.2.7.4 FPTempConvertStr****[Definition]**

VARIANT\_BOOL FPTempConvertStr([in] BSTR TmpData1,[out]BSTR\* TmpData2, [out]  
long \*Size)

**[Usage]**

Convert an Standalone fingerprint template into a Biokey fingerprint template in string form.  
The only difference between this function and FPTempConvert is the data format.

**[Parameter]**

TmpData1

Standalone fingerprint template to be converted

TmpData2

Converted Biokey fingerprint template

Size

Size of the converted Biokey fingerprint template

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

FPTempConvert, FPTempConvertNew, FPTempConvertNewStr

**5.2.7.5 FPTempConvertNew****[Definition]**

VARIANT\_BOOL FPTempConvertNew([in] BYTE\* TmpData1, [out] BYTE\* TmpData2, [out]  
long \*Size)

**[Usage]**

Convert a Biokey fingerprint template into an Standalone fingerprint template. The only  
difference between this function and FPTempConvertNewStr is the data format.

**[Parameter]**

TmpData1

Standalone fingerprint template to be converted into

TmpData2

Converted Standalone fingerprint template

Size

Size of the converted Standalone fingerprint template

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

FPTempConvertNewStr, FPTempConvert, FPTempConvertStr

**5.2.7.6 FPTempConvertNewStr****[Definition]**

VARIANT\_BOOL FPTempConvertNewStr([in] BSTR TmpData1, [out] BSTR\* TmpData2, [out] long \*Size)

**[Usage]**

Convert a Biokey fingerprint template into an Standalone fingerprint template in string form. The only difference between this function and FPTempConvertNew is the data format.

**[Parameter]**

TmpData1

Standalone fingerprint template to be converted into

TmpData2

Converted Standalone fingerprint template

Size

Size of the converted Standalone fingerprint template

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

FPTempConvertNew, FPTempConvert, FPTempConvertStr

**5.2.8 System Data Management Functions****5.2.8.1 ClearKeeperData****[Definition]**

VARIANT\_BOOL ClearKeeperData([in] long dwMachineNumber)

**[Usage]**

Clear all data in the device.

**[Parameter]**

dwMachineNumber

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ClearGLog, ClearSLog

### 5.2.8.2 ClearData

#### [Definition]

VARIANT\_BOOL ClearData([in] LONG dwMachineNumber, [in] LONG DataFlag)

#### [Usage]

Clear the record specified by DataFlag from the device.

#### [Parameter]

dwMachineNumber

Device number

DataFlag

Type of the records to be cleared. The value ranges from 1 to 5. The meanings are as follows:

1. Attendance record
2. Fingerprint template data
3. None
4. Operation record
5. User information

When the value of this parameter is 5, all user data in the device is deleted. Note: All fingerprint templates are also deleted.

#### [Return Value]

Return True if it is successful, or return False.

#### [Related Function]

ClearKeeperData

### 5.2.8.3 GetDataFile

#### [Definition]

VARIANT\_BOOL GetDataFile([in] LONG dwMachineNumber, [in] LONG DataFlag, [in] BSTR FileName)

#### [Usage]

Obtain the specified data file from the device.

#### [Parameter]

dwMachineNumber

Device number

DataFlag

Type of the data file to be obtained

1. Attendance record data file
2. Fingerprint template data file
3. None

4. Operation record data file
5. User information data file
6. SMS data file
7. SMS and user relationship data file
8. Extended user information data file
9. Work code data file

FileName

Name of the obtained data file

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

**5.2.8.4 SendFile**

**[Definition]**

VARIANT\_BOOL SendFile([in] LONG dwMachineNumber,[in] BSTR FileName)

**[Usage]**

Send files to the device, usually to the **/mnt/mtdblock/** directory. For TFT devices, if user pictures or advertisement pictures are sent, the files should be named in the following formats and automatically moved to corresponding directories.

Naming of advertisement pictures: prefix "ad" + a numeral ranging from 1 to 20 + suffix ".jpg", for example, **ad\_4.jpg**

Naming of user pictures: user ID + ".jpg", for example, **1.jpg**

**[Parameter]**

dwMachineNumber

Device number

FileName

Name of the file to be sent

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

ReadFile

**5.2.8.5 RefreshData**

**[Definition]**

VARIANT\_BOOL RefreshData([in] long dwMachineNumber)

**[Usage]**

Refresh the data in the device. This function is usually called after user information or fingerprints are uploaded. In this way, the modifications take effect immediately.



**[Parameter]**

dwMachineNumber  
Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

## 5.3 Access Control Functions (Time Zone, Group, Open Door Combination)

### 5.3.1 GetUserGroup

**[Definition]**

VARIANT\_BOOL GetUserGroup([in] long dwMachineNumber, [in] long dwEnrollNumber, [out] long \*UserGrp)

**[Usage]**

Obtain the number of the group to which a specified user belongs.

**[Parameter]**

dwMachineNumber  
Device number  
dwEnrollNumber  
User ID  
UserGrp  
Group number of the user described by dwEnrollNumber. The value ranges from 1 to 5.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetUserGroup

### 5.3.2 SetUserGroup

**[Definition]**

VARIANT\_BOOL SetUserGroup([in] long dwMachineNumber, [in] long dwEnrollNumber, [in] long UserGrp)

**[Usage]**

Set the group to which a specified user belongs.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

UserGrpl

Group number. The value ranges from 1 to 5.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetUserGroup

### 5.3.3 GetTZInfo

**[Definition]**

VARIANT\_BOOL GetTZInfo([in] long dwMachineNumber, [in] long TZIndex, [out] BSTR \*TZ)

**[Usage]**

Obtain the information of the Time Zone with the specified number.

**[Parameter]**

dwMachineNumber

Device number

TZIndex

Time Zone index

TZ

Time Zone with the index described by TZIndex. Every eight bits represent a Time Zone, in format of hhmmhhmm. For example, 101112230000235900002359000023590000235900002359 indicates a Time Zone covering the whole day from Monday to Saturday and from 10:11 to 12:23 of Sunday.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetTZInfo

### 5.3.4 SetTZInfo

**[Definition]**

VARIANT\_BOOL SetTZInfo([in] long dwMachineNumber, [in] long TZIndex, [in] BSTR TZ)

**[Usage]**

Set the information of the Time Zone with the specified number.

**[Parameter]**

dwMachineNumber

Device number

TZIndex

Time Zone index

TZ

Time Zone to be set. Every eight bits represent a Time Zone, in format of hhmmhhmm. For example, 101112230000235900002359000023590000235900002359 indicates a Time Zone covering the whole day from Monday to Saturday and from 10:11 to 12:23 of Sunday.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetTZInfo

### 5.3.5 SSR\_SetUnLockGroup

**[Definition]**

VARIANT\_BOOL SSR\_SetUnLockGroup([in] LONG dwMachineNumber, [in] LONG CombNo, [in] LONG Group1, [in] LONG Group2, [in] LONG Group3, [in] LONG Group4, [in] LONG Group5)

**[Usage]**

Set the open door combination.

**[Parameter]**

dwMachineNumber

Device number

CombNo

Combination number. The value ranges from 1 to 10, that is, the device supports a maximum of 10 open door combinations.

Group1, Group2, Group3, Group4, Group5

Number of the group in the open door combination. Each open door combination contains five group numbers. Each group number ranges from 1 to 99. For example, SSR\_SetUnLockGroup (1, 1, 2, 23, 14, 0, 56) means that the open is opened only when members of groups 2, 23, 14, and 56 pass verification.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SSR\_GetUnLockGroup

### 5.3.6 SSR\_GetUnLockGroup

**[Definition]**

VARIANT\_BOOL SSR\_GetUnLockGroup([in] LONG dwMachineNumber, [in] LONG CombNo, [out] LONG\* Group1, [out] LONG\* Group2, [out] LONG\* Group3, [out] LONG\* Group4, [out] LONG\* Group5)

**[Usage]**

Obtain the open door combination by open door combination number.

**[Parameter]**

dwMachineNumber

Device number

CombNo

Combination number. The value ranges from 1 to 10.

Group1, Group2, Group3, Group4, Group5

Specific combinations of specified open door groups, that is, which groups are contained in a combination. The group number is returned. Each combination contains a maximum of five groups. Each group number ranges from 1 to 99.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SSR\_SetUnLockGroup

### 5.3.7 SSR\_SetGroupTZ

**[Definition]**

VARIANT\_BOOL SSR\_SetGroupTZ([in] LONG dwMachineNumber, [in] LONG GroupNo, [in] LONG Tz1, [in] LONG Tz2, [in] LONG Tz3, [in] LONG VaildHoliday, [in] LONG VerifyStyle)

**[Usage]**

Set the Time Zone of a group.

**[Parameter]**

dwMachineNumber

Device number

GroupNo

Group number. The value ranges from 1 to -99.

Tz1, Tz2, Tz3

Time Zone index. Each group can contain three Time Zones.

VaildHoliday

Whether holiday settings are valid

VerifyStyle

Verification mode of a group. Meanings: 0(FP/PW/RF), 1(FP), 2(PIN), 3(PW), 4(RF), 5(FP&RF), 6(FP/PW), 7(FP/RF), 8(PW/RF), 9(PIN&FP), 10(FP&PW), 11(PW&RF), 12(FP&PW&RF), 13(PIN&FP&PW), 14(FP&RF/PIN), 15(FACE), 16(FACE&FP), 17(FACE&PW), 18(FACE&RF), 19(FACE&FP&RF), 20(FACE&FP&PW)

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SSR\_GetGroupTZ

### 5.3.8 SSR\_GetGroupTZ

**[Definition]**

VARIANT\_BOOL SSR\_GetGroupTZ([in] LONG dwMachineNumber, [in] LONG GroupNo, [out] LONG\* Tz1, [out] LONG\* Tz2, [out] LONG\* Tz3, LONG\* VaildHoliday, LONG\* VerifyStyle)

**[Usage]**

Obtain the Time Zones of a group.

**[Parameter]**

dwMachineNumber

Number of the fingerprint device

GroupNo

Group number. The value ranges from 1 to 99.

Tz1 , Tz2, Tz3

Indexes of the three Time Zones of a specified combination. Each group number ranges from 1 to 50.

VaildHoliday

Whether holiday settings are valid. 1: valid, 0: invalid

VerifyStyle

Verification mode of the fingerprint device. Values: 0(FP/PW/RF), 1(FP), 2(PIN), 3(PW), 4(RF), 5(FP&RF), 6(FP/PW), 7(FP/RF), 8(PW/RF), 9(PIN&FP), 10(FP&PW), 11(PW&RF), 12(FP&PW&RF), 13(PIN&FP&PW), 14(FP&RF/PIN), 15(FACE), 16(FACE&FP), 17(FACE&PW), 18(FACE&RF), 19(FACE&FP&RF), 20(FACE&FP&PW)

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SSR\_SetGroupTZ

### 5.3.9 GetUserTZs

#### [Definition]

VARIANT\_BOOL GetUserTZs([in] long dwMachineNumber, [in] long UserID, [out] long \*TZs)

#### [Usage]

Obtain the Time Zone setting of a user. Each user has three Time Zones. The only difference between this function and GetUserTZStr is the format of returned Time Zone index.

#### [Parameter]

dwMachineNumber

Device number

UserID

User ID

TZs

Open door Time Zone of a user. The TZs pointers have three values that store indexes of three Time Zones respectively. The indexes of three Time Zones can be obtained by using TZs[0], TZs[1], and TZs[2] respectively.

#### [Return Value]

Return True if it is successful, or return False.

#### [Related Function]

SetUserTZs, SetUserTZStr

### 5.3.10 SetUserTZs

#### [Definition]

VARIANT\_BOOL SetUserTZs([in] long dwMachineNumber, [in] long UserID, [in] long \*TZs)

#### [Usage]

Set the Time Zone of a user. A maximum of three Time Zones can be set for each user. The difference between this function and SetUserTZStr is the format of imported Time Zone index.

#### [Parameter]

dwMachineNumber

Device number

UserID

User ID

TZs

Time Zone index. When TZs[3] is 0, the group setting is used. When TZs[3] is not 0, a Time Zone number set by the user can be specified.

This parameter is a long pointer. Indexes of three Time Zones can be imported respectively by TZs[0], TZs[1], and TZs[2].

#### [Return Value]

Return True if it is successful, or return False.

**[Related Function]**

GetUserTZs, GetUserTZStr

**5.3.11 GetUserTZStr****[Definition]**

VARIANT\_BOOL GetUserTZStr([in] long dwMachineNumber, [in] long UserID, [out] BSTR \*TZs)

**[Usage]**

Obtain the Time Zones of a user. The only difference between this function and GetUserTZs is the format of returned Time Zone index.

**[Parameter]**

dwMachineNumber

Device number

UserID

User ID

TZs

Unlock Time Zone of a user. The formats are as follows:

TFT access control devices: X1:X2:X3:X4. X4 represents whether to use the group Time Zone. If the value is 0, the group Time Zone is used; if the value is **1**, the group Time Zone is not used, that is, the personal Time Zone is used. X1, X2, and X3 represent the indexes of the used Time Zones. For example, if user A uses customized Time Zones 1 and 2, the fingerprint device returns "1:2:0:1".

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetUserTZStr, GetUserTZs

**5.3.12 SetUserTZStr****[Definition]**

VARIANT\_BOOL SetUserTZStr([in] long dwMachineNumber, [in] long UserID, [in] BSTR TZs)

**[Usage]**

Set the Time Zones of a user. Time Zones are separated by ":". The only difference between this function and SetUserTZs is the format of imported Time Zone index.

**[Parameter]**

dwMachineNumber

Device number

UserID

User ID

TZs

See GetUserTZStr.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetUserTZStr, SetUserTZs

### 5.3.13 ACUnlock

**[Definition]**

VARIANT\_BOOL ACUnlock([in] long dwMachineNumber, [in] long Delay)

**[Usage]**

Open the door, enable the open door controller to output a unlock signal, and close the door after 10-second delay.

**[Parameter]**

dwMachineNumber

Device number

Delay

Open door delay

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

### 5.3.14 GetACFun

**[Definition]**

VARIANT\_BOOL GetACFun([out] long\* ACFun)

**[Usage]**

Obtain whether the device has the access control function.

**[Parameter]**

ACFun

Flag of the access control function of the device. 0: no access control, 15: access control available

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

### 5.3.15 GetDoorState



**[Definition]**

VARIANT\_BOOL GetDoorState([in] LONG MachineNumber, [out] LONG\* State)

**[Usage]**

Obtain the current door state. 1: opened, 0: closed. It is Deprecated, please refer to OnDoor Event.

**[Parameter]**

MachineNumber

Device number

State

Door state

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

### 5.3.16 UseGroupTimeZone

**[Definition]**

VARIANT\_BOOL UseGroupTimeZone(void)

**[Usage]**

Determine whether a user uses the group Time Zone setting. This function must be used together with GetUserTZs or GetUserTZStr to ensure the correct return values. That is, use GetUserTZs or GetUserTZStr to obtain the Time Zone settings of the specified user, and then use UseGroupTimeZone to determine whether the user uses the group Time Zone setting.

**[Parameter]**

None

**[Return Value]**

Return True if the group Time Zone is used, or return False.

**[Related Function]**

GetUserTZs, GetUserTZStr

## 5.4 Device Management Functions

### 5.4.1 IsTFTMachine

#### [Definition]

VARIANT\_BOOL IsTFTMachine([in] LONG dwMachineNumber)

#### [Usage]

Determine whether the current device is a TFT device.

#### [Parameter]

dwMachineNumber

Device number

#### [Return Value]

Return True if the current device is a TFT device. Return False if the current device is a black & white device.

#### [Related Function]

None

### 5.4.2 GetDeviceStatus

#### [Definition]

VARIANT\_BOOL GetDeviceStatus( [in] long dwMachineNumber, [in] long dwStatus, [out] long\* dwValue)

#### [Usage]

Obtain the data storage status of the device, for example, number of administrators and number of current users.

#### [Parameter]

dwMachineNumber

Device number

dwStatus

Data to be obtained. The value ranges from 1 to 22. Values:

- 1 Number of administrators
- 2 Number of registered users
- 3 Number of fingerprint templates in the device
- 4 Number of passwords
- 5 Number of operation records
- 6 Number of attendance records
- 7 Fingerprint template capacity

- 8 User capacity
  - 9 Attendance record capacity
  - 10 Residual fingerprint template capacity
  - 11 Residual user capacity
  - 12 Residual attendance record capacity
  - 21 Number of faces
  - 22 Face capacity
- Returned 0 in other cases.

dwValue

Value of dwStatus

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

### 5.4.3 GetDeviceInfo

**[Definition]**

VARIANT\_BOOL GetDeviceInfo( [in] long dwMachineNumber, [in] long dwInfo, [out] long\* dwValue)

**[Usage]**

Obtain related information of the device, such as language and baud rate.

**[Parameter]**

dwMachineNumber

Device number

dwInfo

Type of the information to be obtained. The value ranges from 1 to 68 (except for 65).  
Values:

- 1 Maximum number of administrators. Generally, return 500.
- 2 Device number
- 3 Language

The return values received by dwValue are as follows:

- 0 Language with suffix "E", usually representing English
- 1 Others
- 2 Language with suffix "T", usually representing traditional Chinese
- 3 Language with suffix "L", usually representing Thai language

- 4 Idle duration (minutes). That is, the device enters standby state or is powered off after keeping idle for a period specified by this value.
- 5 Lock duration, namely, driver lock duration
- 6 Number of attendance record alarms. That is, the device reports an alarm when the number of attendance records reaches this value.
- 7 Number of operation record alarms. That is, the device reports an alarm when the number of operation records reaches this value.
- 8 Repeated record time, that is, the minimum interval of opening the same attendance record by the same user
- 9 Baud rate in RS232/RS485 communication

The return values received by dwValue are as follows:

- 0 1200bps
- 1 2400
- 2 4800
- 3 9600
- 4 19200
- 5 38400
- 6 57600
- Others: 115200
- 10 Parity check. Generally, return 0.
- 11 Stop bit. Generally, return 0.
- 12 Date separator. Generally, return 1.
- 13 Whether network function is enabled. Values: 1: enabled, 0: disabled
- 14 Whether RS232 is enabled
- 15 Whether RS485 is enabled
- 16 Whether voice function is supported
- 17 Whether high-speed comparison is performed
- 18 Idle mode, that is, the state that the device enters after idle period. Values: 87: power-off, 88: hibernate
- 19 Automatic power-off time point. The default value is 255, that is, the device does not power off automatically.
- 20 Automatic power-on time point. The default value is 255, that is, the device does not power on automatically.
- 21 Automatic hibernation time point. The default value is 255, that is, the device does not hibernate automatically.
- 22 Automatic ring time point. The default value is 65535, that is, the device does not ring automatically.

- 23 1:N match threshold
- 24 Match threshold during registration
- 25 1:1 match threshold
- 26 Whether to display match score during verification
- 27 Number of concurrent unlock users
- 82 Verify only card number
- 29 Network speed
  - The return values received by dwValue are as follows:
  - 1 100M-H
  - 4 10M-F
  - 5 100M-F
  - 8 AUTO
  - Others: 10M-H
- 30 Whether the card must be registered
- 31 Waiting time before the device automatically returns when there is temporarily no operation
- 32 Waiting time before the device automatically returns when no response is returned after the PIN is input
- 33 Waiting time before the device automatically returns when there is response after entering the menu
- 34 Time format
- 35 Whether 1:1 match must be used
- 36—40 Automatic ring time points 2, 3, 4, 5, and 6. The default values are **65535**, that is, the device does not ring automatically.
- 41—56 Automatic state change time points 1 to 16. The default values are **-1**, that is, the device does not change state automatically.
- 57 Wiegand failure ID
- 58 Wiegand threaten ID
- 59 Wiegand region-position code
- 60 Wiegand output pulse width
- 61 Wiegand output pulse interval
- 62 Start sector of Mifare card for storing fingerprints
- 63 Total sectors of Mifare card for storing fingerprints
- 64 Number of fingerprints stored on Mifare card
- 66 Whether to display attendance state
- 67,68 Not supported

8999 In this case, dwValue is used as both input and output parameters. As the input parameter, dwValue represent the name of another option to be obtained. As the output parameter, dwValue represents the value of the option (in this case, dwValue is similar to GetSysOption).

Note: The return values of the preceding time points are numerals. To convert the numeral into the time point, convert the value into a binary numeral where the lowest eight bits represent minute and the highest bits represent hour. For example, if the return value is 2860, it can be converted into 101100101100 in binary, of which the lowest eight bits 00101100 (that is, 44) and the highest eight bits are 00001011 (that is, 11), that is, the actual time point is 11:44.

dwValue

Value described by dwInfo

#### **[Return Value]**

Return True if it is successful, or return False.

#### **[Related Function]**

SetDeviceInfo

### **5.4.4 SetDeviceInfo**

#### **[Definition]**

VARIANT\_BOOL SetDeviceInfo([in] long dwMachineNumber, [in] long dwInfo, [in] long dwValue)

#### **[Usage]**

Set the related information of the device, such as language and repeated record time.

#### **[Parameter]**

dwMachineNumber

Device number

dwInfo

Type of the information to be set. The value ranges from 1 to 20. For the meanings of values, see GetDeviceInfo.

dwValue

Value of the information described by dwInfo

#### **[Return Value]**

Return True if it is successful, or return False.

#### **[Related Function]**

GetDeviceInfo

### **5.4.5 SetDeviceTime**

#### **[Definition]**

VARIANT\_BOOL SetDeviceTime([in] long dwMachineNumber)

**[Usage]**

Set the local computer time to the device time. To set the specified time, see SetDeviceTime2.

**[Parameter]**

dwMachineNumber  
Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetDeviceTime2, GetDeviceTime

## 5.4.6 SetDeviceTime2

**[Definition]**

VARIANT\_BOOL SetDeviceTime2([in] LONG dwMachineNumber, [in] LONG dwYear, [in] LONG dwMonth, [in] LONG dwDay, [in] LONG dwHour, [in] LONG dwMinute, [in] LONG dwSecond)

**[Usage]**

Set the device time (or specify the time).

**[Parameter]**

dwMachineNumber  
Device number  
dwYear, dwMonth, dwDay, dwHour, dwMinute, dwSecond  
Date and time to be set

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetDeviceTime, GetDeviceTime

## 5.4.7 GetDeviceTime

**[Definition]**

VARIANT\_BOOL GetDeviceTime([in] long dwMachineNumber, [out] long\* dwYear, [out] long\* dwMonth, [out] long\* dwDay, [out] long\* dwHour, [out] long\* dwMinute, [out] long\* dwSecond)

**[Usage]**

Obtain the device time.

**[Parameter]**

dwMachineNumber

Device number

dwYear, dwMonth, dwDay, dwHour, dwMinute, dwSecond

Long pointers pointing to variables. The values are the date and time of the device.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetDeviceTime, SetDeviceTime2

## 5.4.8 GetSerialNumber

**[Definition]**

VARIANT\_BOOL GetSerialNumber( [in] long dwMachineNumber, [out] BSTR\* dwSerialNumber)

**[Usage]**

Obtain the serial number of the device.

**[Parameter]**

dwMachineNumber

Device number

dwSerialNumber

Serial number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

## 5.4.9 GetProductCode

**[Definition]**

VARIANT\_BOOL GetProductCode( [in] long dwMachineNumber, [out] BSTR\* lpszProductCode)

**[Usage]**

Obtain device name.

**[Parameter]**

dwMachineNumber

Device number

lpszProductCode

Device name



**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

### 5.4.10 GetFirmwareVersion

**[Definition]**

VARIANT\_BOOL GetFirmwareVersion( [in] long dwMachineNumber, [out] BSTR\* strVersion)

**[Usage]**

Obtain the firmware version of the device.

**[Parameter]**

dwMachineNumber

Device number

strVersion

Firmware version

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

### 5.4.11 GetSDKVersion

**[Definition]**

VARIANT\_BOOL GetSDKVersion( [out] BSTR\* strVersion)

**[Usage]**

Obtain the SDK version.

**[Parameter]**

strVersion

SDK version

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

### 5.4.12 GetDeviceIP

**[Definition]**

VARIANT\_BOOL GetDeviceIP( [in] long dwMachineNumber, [out] BSTR \*IPAddr)

**[Usage]**

Obtain the IP address of the device.

**[Parameter]**

dwMachineNumber

Device number

IPAddr

IP address

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetDeviceIP

### 5.4.13 SetDeviceIP

**[Definition]**

VARIANT\_BOOL SetDeviceIP( [in] long dwMachineNumber, [in] BSTR IPAddr)

**[Usage]**

Set the IP address of the device.

**[Parameter]**

dwMachineNumber

Device number

IPAddr

IP address

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetDeviceIP

### 5.4.14 GetDeviceMAC

**[Definition]**

VARIANT\_BOOL GetDeviceMAC( [in] LONG dwMachineNumber, [out] BSTR \*sMAC)

**[Usage]**

Obtain the MAC address of the device.

**[Parameter]**

dwMachineNumber

Device number

sMAC

MAC address

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetDeviceMAC

## 5.4.15 SetDeviceMAC

**[Definition]**

VARIANT\_BOOL SetDeviceMAC( [in] LONG dwMachineNumber, [in] BSTR sMAC)

**[Usage]**

Set the MAC address of the device.

**[Parameter]**

dwMachineNumber

Device number

sMAC

MAC address

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetDeviceMAC

## 5.4.16 GetWiegandFmt

**[Definition]**

VARIANT\_BOOL GetWiegandFmt( [in] LONG dwMachineNumber, [out] BSTR \*sWiegandFmt)

**[Usage]**

Obtain Wiegand format of the device.

**[Parameter]**

dwMachineNumber

Device number

sWiegandFmt

Wiegand format

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetWiegandFmt

### 5.4.17 SetWiegandFmt

**[Definition]**

VARIANT\_BOOL SetWiegandFmt([in] LONG dwMachineNumber, [in] BSTR sWiegandFmt)

**[Usage]**

Set Wiegand format of the device.

**[Parameter]**

dwMachineNumber

Device number

sWiegandFmt

Wiegand format

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetWiegandFmt

### 5.4.18 GetCardFun

**[Definition]**

VARIANT\_BOOL GetCardFun( [in] LONG dwMachineNumber, [in] LONG\* CardFun)

**[Usage]**

Obtain whether the device supports the RF card.

**[Parameter]**

dwMachineNumber

Device number

CardFun

Values: 1: The device supports only RF card verification. 2: The device supports both RF card verification and fingerprint verification. 0: The device does not support RF card verification.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

### 5.4.19 SetDeviceCommPwd

**[Definition]**

VARIANT\_BOOL SetDeviceCommPwd( [in] LONG dwMachineNumber, [in] LONG CommKey)

**[Usage]**

Set the communication password of the device. The communication password is stored in the device.

**[Parameter]**

dwMachineNumber

Device number

CommKey

Communication password

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetCommPassword

### 5.4.20 SetCommPassword

**[Definition]**

VARIANT\_BOOL SetCommPassword( [in] long CommKey)

**[Usage]**

Set the communication password of the PC. A connection can be set up only when the PC and the device use the same communication password.

**[Parameter]**

CommKey

Communication password

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

SetDeviceCommPwd

### 5.4.21 QueryState

**[Definition]**

VARIANT\_BOOL QueryState([out] LONG \*State)

**[Usage]**

Query the current state of the device.

**[Parameter]**

State

Current state of the device. Values are as follows:

0   Waiting

- 1 Registering a fingerprint
- 2 Identifying a fingerprint
- 3 Accessing menu
- 4 Busy (doing other tasks)
- 5 Waiting for writing data into card

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

## 5.4.22 GetVendor

**[Definition]**

VARIANT\_BOOL GetVendor([in] BSTR\* strVendor)

**[Usage]**

Obtain the vendor name of the device.

**[Parameter]**

strVendor

Vendor name of the device

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

## 5.4.23 GetDeviceStrInfo

**[Definition]**

VARIANT\_BOOL GetDeviceStrInfo([in] LONG dwMachineNumber, [in] LONG dwInfo, [out] BSTR\* Value)

**[Usage]**

Obtain the manufacturing time of the device.

**[Parameter]**

dwMachineNumber

Device number

dwInfo

This parameter can be set to 1 only.

Value

Manufacturing time of the device

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetDeviceInfo

## 5.4.24 GetPlatform

**[Definition]**

VARIANT\_BOOL GetPlatform([in] LONG dwMachineNumber, [out] BSTR\* Platform)

**[Usage]**

Obtain the platform name of the device.

**[Parameter]**

dwMachineNumber

Device number

Platform

Platform name

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

## 5.4.25 ReadAOptions

**[Definition]**

VARIANT\_BOOL ReadAOptions([in] BSTR AOption, [out] BSTR\* AValue)

**[Usage]**

Read the values of specified configuration parameters from the device. The parameters beginning with "~" are skipped.

**[Parameter]**

Aoption

Parameter name

Avalue

Value of the parameter described by Aoption

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

GetSysOption

## 5.4.26 GetSysOption

### [Definition]

VARIANT\_BOOL    GetSysOption([in] LONG dwMachineNumber, [in] BSTR Option, [out] BSTR\* Value)

### [Usage]

Obtain the parameters from the device. Note: This function can be used to obtain the algorithm version used by the device.

### [Parameter]

dwMachineNumber

Device number

Option

Parameter name.

Value

Value of the parameter described by Option

### [Return Value]

Return True if it is successful, or return False.

### [Related Function]

SetSysOption

## 5.4.27 SetSysOption

### [Definition]

VARIANT\_BOOL    SetSysOption([in] LONG dwMachineNumber, [in] BSTR Option, [in] BSTR Value)

### [Usage]

Configure the parameters in the device.

### [Parameter]

dwMachineNumber

Device number

Option

Name of the parameter to be set

Value

Value of the parameter described by Option

### [Return Value]

Return True if it is successful, or return False.

### [Related Function]

GetSysOption



## 5.5 Others

### 5.5.1 Device Control Functions

#### 5.5.1.1 ClearAdministrators

**[Definition]**

VARIANT\_BOOL ClearAdministrators([in] long dwMachineNumber)

**[Usage]**

Clear all administrator privileges from the device.

**[Parameter]**

dwMachineNumber

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

#### 5.5.1.2 EnableDevice

**[Definition]**

VARIANT\_BOOL EnableDevice([in] long dwMachineNumber, [in] VARIANT\_BOOL bFlag)

**[Usage]**

Enable or disable the device. If the device is disabled, the fingerprint sensor, keypad, card modules, etc. are disabled.

**[Parameter]**

dwMachineNumber

Device number

bFlag

User enable flag. 1: Enabled. 0: Disabled.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

#### 5.5.1.3 EnableClock

**[Definition]**

VARIANT\_BOOL EnableClock([in] LONG Enabled)

**[Usage]**

Enable or disable the clock display with colon ":". If enabled, the device clock is displayed with a colon and synchronized to the main interface. If disabled, the clock is displayed without a colon.

**[Parameter]**

Enabled

Display control. 1: Enabled. 0: Disabled.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

#### **5.5.1.4 DisableDeviceWithTimeOut**

**[Definition]**

VARIANT\_BOOL DisableDeviceWithTimeOut( [in] LONG dwMachineNumber, [in] LONG TimeOutSec)

**[Usage]**

Disable the device for a period of time.

**[Parameter]**

dwMachineNumber

Device number

TimeOutSec

Duration of disabling the device

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

EnableDevice

#### **5.5.1.5 PowerOffDevice**

**[Definition]**

VARIANT\_BOOL PowerOffDevice([in] long dwMachineNumber)

**[Usage]**

Power off the device.

**[Parameter]**

dwMachineNumber

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

**5.5.1.6 RestartDevice****[Definition]**

VARIANT\_BOOL RestartDevice([in] LONG dwMachineNumber)

**[Usage]**

Restart the device.

**[Parameter]**

dwMachineNumber

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

PowerOffDevice

**5.5.1.7 SleepDevice****[Definition]**

VARIANT\_BOOL SleepDevice([in] long dwMachineNumber)

**[Usage]**

Enable the device to enter hibernation mode.

**[Parameter]**

dwMachineNumber

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

RestartDevice, PowerOffDevice, ResumeDevice, SuspendDevice

**5.5.2 Card Operation Functions****5.5.2.1 WriteCard****[Definition]**

VARIANT\_BOOL WriteCard([in] LONG dwMachineNumber, [in] LONG dwEnrollNumber, [in] LONG dwFingerIndex1, [in] BYTE\* TmpData1, [in] LONG dwFingerIndex2, [in] BYTE\* TmpData2, [in] LONG dwFingerIndex3, [in] BYTE\* TmpData3, [in] LONG dwFingerIndex4, [in] BYTE\* TmpData4,)

**[Usage]**

Write the information and fingerprint template of a specified user into the MF card. After this function is called, the MF card must be verified by the device.

**[Parameter]**

dwMachineNumber

Device number

dwEnrollNumber

User ID

dwFingerIndex1, dwFingerIndex2, dwFingerIndex3, dwFingerIndex4

Index of fingerprint (0-3)

TmpData1, TmpData2, TmpData3, TmpData4

Fingerprint templates corresponding to fingerprints. TmpData1 cannot be null.

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

EmptyCard

### 5.5.2.2 EmptyCard

**[Definition]**

VARIANT\_BOOL EmptyCard([in] LONG dwMachineNumber)

**[Usage]**

Clear the data from the MF card.

**[Parameter]**

dwMachineNumber

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

WriteCard

### 5.5.3 Others

#### 5.5.3.1 GetLastError

**[Definition]**

GetLastError( [out] long\* dwErrorCode)

**[Usage]**

Obtain the last error information.

**[Parameter]**

dwErrorCode

Error code returned. Values are as follows:

-100	Operation failed or data not exist
-10	Transmitted data length is incorrect
-5	Data already exists
-4	Space is not enough
-3	Error size
-2	Error in file read/write
-1	SDK is not initialized and needs to be reconnected
0	Data not found or data repeated
1	Operation is correct
4	Parameter is incorrect
101	Error in allocating buffer

**[Return Value]**

None

**[Related Function]**

Return True if it is successful, or return False.

### 5.5.3.2 GetHIDEventCardNumAsStr

**[Definition]**

VARIANT\_BOOL GetHIDEventCardNumAsStr([out] BSTR\* strHIDEventCardNum)

**[Usage]**

Obtain the number of the card that is latest used.

**[Parameter]**

strHIDEventCardNum

Number of the card lately used

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

OnHIDNum

### 5.5.3.3 CaptureImage

**[Definition]**

VARIANT\_BOOL CaptureImage([in] VARIANT\_BOOL FullImage, [in] LONG \*Width, [in] LONG \*Height, [in] BYTE \*Image, [in] BSTR ImageFile)

**[Usage]**

Capture the image of the fingerprint on the fingerprint sensor.

**[Parameter]**

FullImage

Whether to capture the entire image. Return True if the device captures the whole image. Return False if the device captures only the fingerprint.

Width

Width of the image to be captured

Height

Height of the image to be captured

Image

Binary fingerprint image

ImageFile

Storage name of the specified fingerprint image to be captured (including the storage path)

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

**5.5.3.4 UpdateFirmware****[Definition]**

VARIANT\_BOOL UpdateFirmware([in] BSTR FirmwareFile)

**[Usage]**

Upgrade the firmware. To use this function, you need to obtain corresponding firmware from technical engineers of ZKSoftware.

**[Parameter]**

FirmwareFile

File name of the firmware to be upgraded (including the path)

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

None

**5.5.3.5 BeginBatchUpdate****[Definition]**

VARIANT\_BOOL BeginBatchUpdate([in] LONG dwMachineNumber, [in] LONG UpdateFlag)

**[Usage]**

Start uploading data in batches. For example, if you call this function before uploading data such as user templates and user information, the SDK stores the data temporarily in the buffer during upload. Then, you can run BatchUpdate to import temporary data into the device.

**[Parameter]**

dwMachineNumber

Device number

UpdateFlag

Fingerprint overwrite flag, that is, when a user fingerprint template is uploaded, if the fingerprint index has been specified for an existing fingerprint, the device prompts whether to overwrite the existing fingerprint template. 1: Forcibly overwrite, 0: Not overwrite

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

BatchUpdate, CancelBatchUpdate

### 5.5.3.6 BatchUpdate

**[Definition]**

VARIANT\_BOOL BatchUpdate([in] LONG dwMachineNumber)

**[Usage]**

Start uploading data in batches. Generally, this function is used only after BeginBatchUpdate is used to upload related data.

**[Parameter]**

dwMachineNumber

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

BeginBatchUpdate, CancelBatchUpdate

### 5.5.3.7 CancelBatchUpdate

**[Definition]**

VARIANT\_BOOL CancelBatchUpdate([in] LONG dwMachineNumber)

**[Usage]**

Cancel uploading data in batches. Generally, this function can be used to release the buffer reserved for batch upload after BeginBatchUpdate and before BatchUpdate.

**[Parameter]**

dwMachineNumber

Device number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

BeginBatchUpdate, BatchUpdate

### 5.5.3.8 PlayVoice

**[Definition]**

VARIANT\_BOOL PlayVoice([in] LONG Position, [in] LONG Length)

**[Usage]**

Play tones with the specified consecutive numbers. Tone numbers are determined by the device. You can view the tone numbers in Voice Test menu of the device. Generally, the values range from 0 to 11.

**[Parameter]**

Position

Start tone number

Length

End tone number

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

PlayVoiceByIndex

### 5.5.3.9 PlayVoiceByIndex

**[Definition]**

VARIANT\_BOOL PlayVoiceByIndex([in] LONG Index)

**[Usage]**

Play tones with the specified numbers. Tone numbers are determined by the device. You can view the tone numbers in Voice Test menu of the device. Generally, the values range from 0 to 11.

**[Parameter]**

Index

Number of the tone to be played

**[Return Value]**

Return True if it is successful, or return False.

**[Related Function]**

PlayVoice



## 6 FAQs

### 6.1 How to Download Attendance Records?

First, use `ReadGeneralLogData` to read all attendance records and write them into the memory. Then, use `GetGeneralLogData` repeatedly to obtain attendance records. When `GetGeneralLogData` returns `False`, it means that all attendance records are obtained. Then, you can write the obtained records into database or display them in other forms to finish downloading. You can follow the same steps to down operation records.

### 6.2 How to Create a User Online?

First, use `SetUserInfo` to write user information (such as enrollment number, password, and name) into the device. Then, use `SetUserTmpStr/SetUerTmp/SetEnrollDataStr/SetEnrollData` to set fingerprint templates for the user. This method improves enrollment efficiency and is suitable when user information has been collected and stored in media such as database.

To upload user information and corresponding fingerprint templates in batches, use `BeginBatchUpdata` or `SetUserInfo` together with `SetUserInfo`, `BarchUpdata`, `EnableDevice`, or `RefreshData`. For details, see demo program.

### 6.3 How to Import or Download Data from USB Disk?

Many existing offline products support data download from USB disks. Many customers are concerned about data formats of USB disks. As the downloaded data formats are complex, ZKSoftware provides a tool for importing data from USB disks to database. Database is open for customers to download data. In addition, ZKSoftware also provides examples on how to process data files (\*.dat, etc.) collected from USB disks and how to write the data into specified data files. All structs adopt 1 byte alignment mode.

Data in USB disks include user information, fingerprint templates, face templates, attendance records, and short messages. Detailed data structures are used in demo program. They are described briefly below:

User data structure:

```
typedef struct _User_{
    U16 PIN; //Internal number of a user
    U8 Privilege;
    char Password[8];
    char Name[24];
    U8 Card[4];
    U8 Group;
```

```

    U16 TimeZones[4];
    char PIN2[24];    //User ID
}

```

Data structure of template.fp10:

```

typedef struct _Template_{
    U16 Size;                //entire structure data size
    U16 PIN;                 //user ID
    BYTE FingerID;           //fingerprint index
    BYTE Valid;              //flag
    BYTE *Template; //template
}

```

Attendance record,

```

struct _AttLog_{
    U16 PIN;
    U8 PIN2[24];
    BYTE verified;
    time_t time_second;
    BYTE status;
    U32 workcode;
    BYTE reserved[4];
}

```

Exported as a text file:

attlog.dat format      explanation:

segment:

BadgeNumber(employee number),

checktime,

DeviceID,

checktype(check status),

VerifyCode(verification ways:password or fingerprint or other)

Workcode

There is an Ascii code #9(Tab) between each segment. When development, move to the segment value you want to choose by "Tab".

If the device has the output data protection function, the serial number of the current device is displayed in first line and the hash value in the last line of the file to which attendance records are exported from USB disk.

Data structure of SMS

```

typedef struct _SMS_{
    BYTE Tag;        //category
    U16 ID;           //data content flag. 0 indicates that the record is invalid.
    U16 ValidMinutes; //valid minutes. 0 indicates that the record is permanently valid.
    U16 Reserved;
    U32 StartTime;   //start time
    BYTE Content[MAX_SMS_CONTENT_SIZE+1]; //short message content
} // MAX_SMS_CONTENT_SIZE    60 Bytes

```

Data structure between SMS and user pin//user->sms,udata.dat

```

typedef struct _UData_{
    U16 PIN;          //0 indicates that the record is invalid
    U16 SmsID;
}GCC_PACKED TUData, *PUData; //4Bytes

```

Data structure of face template:

```

typedef struct _FaceTmp_{
    U16 Size;//face template size
    U16 PIN;//user ID
    BYTE FaceID;//Face ID
    BYTE Valid;//flag
    U16 Reserve;//reserve
    U32 ActiveTime;
    U32 VfCount;//Verify Count
    BYTE FaceTmp[FACE_TMP_MAXSIZE]
} //FACE_TMP_MAXSIZE=1024*2+512

```

## 6.4 How to Use Biokey to Write the Collected Fingerprint Templates Offline?

When a fingerprint is collected, Biokey usually obtains the fingerprint template during enrollment. For example, the currently enrolled fingerprint template can be obtained via OnEnroll. After obtaining the fingerprint template, Biokey converts it into an Standalone fingerprint template. Then, the template can be written into the device.

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

Use ReadAllUserID to read IDs of all users and write them into memory. Then, use GetAllUserID repeatedly to obtain EnrollNumber of users, and use GetUserInfo to obtain user information. If necessary, you can also use GetUserTmpStr to obtain the fingerprint templates in string form.

## 6.6 How to Connect to the Device?

During connection, the device can be regarded as an independent PC. However, the IP address of the device must match the IP address to be connected. Some devices, for example F4, support serial port connection and network connection. Therefore, during different connections, you need to set the device differently, modify communication mode, and set the controller switch to TCP/IP or RS232/485. Otherwise, the connection may fail. Sometimes, if the device fails to be connected due to busy serial ports, you can restart the program. If the application keeps connecting to the device without being manually disconnected, you can use `DisableDeviceWithTimeOut` to set the automatic disconnection time of the device. If some connections are used to download or modify data via serial ports or network, you can use `EnableDevice` to keep the device working and restore the connections after communication finishes, so as to maintain data consistency and avoid unknown errors.

## 6.7 Password Is Invalid After SSR\_SetUserInfo Is Used.

After `SetUserInfo` is called, Password may be set to null. If so, password verification will fail. To keep the password unchanged when writing user information, use `GetUserInfo` to obtain user password and transmit the password value to the Password parameter of `SetUserInfo` before using `SetUserInfo`.

## 6.8 How to Convert an Online Template into an Offline Template?

Use `FPTempConvertNew` to convert the collected templates into Standalone fingerprint templates. See related descriptions of Biokey SDK for how to obtain the templates collected by Biokey. `FPTempConvertNew` is used to convert binary fingerprint templates. Parameters `temp1` and `temp2` are binary parameters. You can also use `FPTempConvertNewStr` to convert Biokey fingerprint templates of string type into Standalone fingerprint templates.

## 6.9 Demo Program Fails to Connect to the Device.

After the attendance management program is installed, users can connect to the device by using the attendance management program, but cannot connect to the device by using demo program. The reason is that DLL is copied to the directory of the attendance management program but registered in the installation directory during program installation. Generally, SDK loads controls from the system directory. Therefore, if the SDK version in the system directory is different from that in the attendance software directory, conflicts occur. (DLL function addresses of different versions are different, but OCX functions are the same in programming. Therefore, the problem is found only during runtime.)

Caution: The common procedure for registering the SDK in the system is as follows:

1. If an SDK has been already registered in the system, run **regsvr32 /u      zkemkeeper.dll** to unregister the original SDK.
2. Copy all DLLs to the system directory, for example, win2000 is located in winnt\system32.
3. Run `regsvr32 "registration path\zkemkeeper.dll"` to register the SDK.

4. Correctly load controls in development environment (learn the usage of development tool by yourself. Relevant details are omitted here).
5. Try to use the SDK of the same version in development or running environment.

## **6.10 Standalone fingerprint Device Keeps Working After Being Connected.**

After connecting the SDK to the Standalone fingerprint device, use EnableDevice to keep the Standalone fingerprint device working (see EnableDevice), so as to maintain data consistency and avoid unknown errors. After the Standalone fingerprint device is working, the keypad and fingerprint sensor stop working. After communication finishes, disconnect the SDK from the device or use EnableDevice again to restore the Standalone fingerprint device to normal state.

DisableDeviceWithTimeOut is recommended.

## **6.11 Illegal Characters Are Displayed or Screen Display Is Abnormal After Non-English Names or Short Messages Are Uploaded to the Device.**

First, check whether the device supports the specified language. For example, if the current language of the device is English, but an Arabic name is uploaded to the device, the name cannot be displayed normally. If the device already supports the language, but the name still cannot be displayed, use related functions to convert the user name into UTF-8 format (for example, use AnsiToUTF8() in Dephi), and then use SetUserInfo to upload the user name.

## **6.12 Card Management Problems**

How to register a card in the device? How to obtain the user card?

The SDK has the cardnumber parameter. If this parameter is invisible in development environment, use GetStrCardNumber and SetStrCardNumber instead.

For a user enrolled in the device, the card number is a kind of user information. When SetUserInfo is used to set user information, cardnumber is automatically used as the card number and set for the user.

The procedure for registering a card is as follows:

Set cardnumber -> Upload user information

The procedure for obtaining the card number of a user is as follows:

Obtain information of the specified user -> Obtain cardnumber

Note: The card number is internally defined as four unsigned bytes of long type. If VB does not support four unsigned bytes, verification can be started after the last three bytes of the card number are input (if the last three bytes are different from each other).

## 6.13 Firewall or Router Traversal

In most cases, the device to be connected needs to traverse firewalls or routers, and UDP socket and port 4370 are used for SDK communication. Therefore, UDP and port 4370 must be enabled on firewalls or routers. If the device traverses gateways via port mapping, the device can be accessed via port number and IP address of routers. Generally, if UDP and port 4370 are enabled and can be pinged, the device can be connected. Certainly, in the case of data download, network connection must be considered. In addition, some devices that support SOAP ports can be accessed via embedded Web Server and SOAP.

## 6.14 Uploading a Large Capacity of Fingerprints

Large capacity usually means over 1500 fingerprints. Some devices can hold 8000 fingerprints or more. Fingerprint templates must be uploaded in batches. In this mode, the upload is much faster. For how to upload fingerprint templates in batches, see descriptions of batch process function.

## 6.15 Differences between High-speed Upload and Ordinary Upload

In an ordinary upload, each time upload functions (such as SetUserinfo and SetUserTmpStr) are used, the SDK communicates with the device and uploads related data to the device.

In a high-speed upload, BeginBatchUpdata is used to create a temporary buffer to store the data to be uploaded in subsequent operations. Then, BatchUpdata can be used to upload all the data in the buffer to the device at a time. This mode greatly reduces communications between the SDK and the device, and raises the speed of large-capacity upload in particular.

## 6.16 How to Download a Face Template?

1. The transmission mode of face templates is the same as that of ZKFinger10.0 fingerprint templates.
2. One user has about 15 face templates in different angles. Each template consists of 2576 bytes. The third and fourth bytes of each template indicate the ID, corresponding to the first and second bytes of user structure. For devices that support face identification, the last 24 bytes of user structure indicate the user ID. Therefore, the total size of face templates of each user is about 37 KB. You are not recommended to upload or download data via serial ports. When the value of dwFaceIndex is 50, all face templates of a user are uploaded or downloaded.

3. To upload and download ZKFace templates, use the following functions:

```
SetUserFace(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFaceIndex, BYTE* TmpData, LONG TmpLength, VARIANT_BOOL* pVal);
```

```
GetUserFace(LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFaceIndex, BYTE* TmpData, LONG * TmpLength, VARIANT_BOOL* pVal);
```

```
DelUserFace (LONG dwMachineNumber, BSTR dwEnrollNumber, LONG dwFaceIndex, VARIANT_BOOL* pVal);
```

For details, see description of the functions.