SBXPC Reference Manual

V3.14

Beijing Smackbio Technology Co., Ltd.

5/5/2016

Revision History

Important Note:

Smackbio always guarantees backward compatibility of OCX for existing devices, new OCX always embraces all products. Therefore it is strongly recommended to use new OCX.

Rev.	Date	Comments
V3.0 Draft	7/4/2011	Draft version for the unified OCX (supports all the types of machine)
V3.01 Draft	7/18/2011	Draft version. If an interface gets time, then it always gets in second unit. General operation by XML & general event firing by XML added, this means that future modification or addition is forbidden, they must be implemented through these XML interfaces.
V3.02 Draft	8/3/2011	Draft version. Brief description of XML operation is added.
V3.03	11/18/2011	New DLL mode support. See ***5.DLL mode support***.
V3.1	6/8/2012	Obsolete ****_EXT*** functions were removed. These functions never used by customers, so they were removed.
		Multi-device safe, multi-thread safe support. See ***6.Multi-device safe, Multi-thread safe support***.
		"TCP active connection" mode support. See ***7."TCP active connection" mode support***.
		P2P support. See ***8.P2P support***.
		See ***Appendix 2. Miscellaneous Items***
V3.11	8/21/2013	Added a new interface for setting time. See ***3.57.SetDeviceTime1***.
V3.12	12/12/2013	Added face related options. See ***3.3 GetEnrollData,*** ***3.7 DeleteEnrollData,*** ***3.9 GetSuperLogData,*** ***3.11 GetGeneralLogData,*** ***3.16 GetDeviceStatus*** ***3.17 GetDeviceInfo,*** ***3.54 GetDeviceLongInfo***.
V3.13	4/26/2016	Added the Realtime Camera option. see ***3.17 GetDeviceInfo,*** ***3.18 SetDeviceInfo.***
V3.14	5/5/2016	Added the Use Fail Log option. see ***3.17 GetDeviceInfo,*** ***3.18 SetDeviceInfo.***

Contents

1. Brief description

1.1 Support machines

S100 (SB2900), S300 (SB2960), SB3600, SB2910 (EC500)
A30 (AR1003), A50 (AR1005)
F500

1.2 SBXPC

SBXPC reference consists of **SBXPC.OCX** which is a Windows OCX component containing the properties, methods and event handlers as the below. If you want to see more specific machine functionality it is strongly recommended to see "Appendix 1. Specific Machine Functionality" first.

No	Properties/Methods	Description
2 Properties		
1	CommPort (**deprecated**)	The port number of serial COM.
2	Baudrate(**deprecated**)	The baudrate of serial COM.
3	ReadMark	The flag which represents whether set a mark on the last read record.
3 Methods		
1	SetMachineType	Sets the machine type.
2	DotNET	Informs that client development environment is .NET.
3	GetEnrollData	Gets an enrolled data from the machine.
4	GetEnrollData1	Gets an enrolled data from the machine(for .NET users).
5	SetEnrollData	Sets an enrolled data to the machine.
6	SetEnrollData1	Sets an enrolled data to the machine (for .NET users).
7	DeleteEnrollData	Removes an enrolled data from the machine.
8	ReadSuperLogData	Reads the new management records from the machine and stores on memory.
9	GetSuperLogData	Gets a management record from the memory.
10	ReadGeneralLogData	Reads the new attendance records from the machine and stores on memory.
11	GetGeneralLogData	Gets a attendance record from the memory.
12	ReadAllSLogData	Reads all the management records from the device and stores on memory.
13	GetAllSLogData	Gets all the management records from the memory.
14	ReadAllGLogData	Reads all the attendance records from the machine and stores on memory.
15	GetAllGLogData	Gets all the attendance records from the memory.
16	GetDeviceStatus	Gets the current device status from the machine.
17	GetDeviceInfo	Gets the device information from the machine.

18	SetDeviceInfo	Sets the device information to the machine.
19	EnableDevice	Enables(or disables) the machine(in order to get the attendance records etc).
20	EnableUser	Enables(or disables) the machine(in order to get an attendance record of a user).
21	GetDeviceTime	Gets the date and time on the machine.
22	SetDeviceTime	Sets the data and time on the machine.
23	PowerOnAllDevice	Powers on all the machines.
24	PowerOffDevice	Powers off an machine.
25	ModifyPrivilege	Modifies the privilege of a user on the machine.
26	ReadAllUserID	Reads the data of all the enrolled users from the machine and stores on memory.
27	GetAllUserID	Gets the data of all the enrolled users from the memory.
28	GetSerialNumber	Gets the serial number of the machine.
29	GetBackupNumber	Gets the management number of the machine.
30	GetProductCode	Gets the product code of the machine.
31	ClearKeeperData	Clears all the records and enrolled data.
32	EmptyEnrollData	Clears all the enrolled data on the machine.
33	EmptyGeneralLogData	Clears all the attendance(access) records.
34	EmptySuperLogData	Clears all the management records.
35	GetUserName	Gets the name of the specified user from the machine.
36	GetUserName1	Gets the name of a user from the machine(for .NET users).
37	SetUserName	Sets the name of the specified user to the machine.
38	SetUserName1	Sets the name of a user to the machine(for .NET users).
39	GetCompanyName	Gets the company name from the machine.
40	GetCompanyName1	Gets the company name from the machine(for .NET users).
41	SetCompanyName	Sets the company name to the machine.
42	SetCompanyName1	Sets the company name to the machine(for .NET users).
43	GetDoorStatus	Gets the door's status from the machine.
44	SetDoorStatus	Sets the door's status to the machine.
45	GetBellTime	Gets the bell time from the machine.
46	SetBellTime	Sets the bell time to the machine.
47	ConnectSerial	Connects to the machine via the serial port (RS232, RS485) or USB.
48	ConnectTcpip	Connects to the machine via the ethernet(TCP/IP).
49	Disconnect	Disconnects the machine.
50	SetIPAddress(**deprecated**)	Informs the IP address of machine.
51	OpenCommPort(**deprecated**)	Connects the machine with current communication settings.
52	CloseCommPort(**deprecated**)	Disconnects the machine.

53	GetLastError	Gets the last error code.
54	GeneralOperationXML	General operation by XML.
55	GetDeviceLongInfo(**deprecated**)	Gets the device long information from the machine.
56	SetDeviceLongInfo(**deprecated**)	Sets the device long information to the machine.
57	ModifyDuressFP(**deprecated**)	Modifies the fingerprint duress status on the machine.
58	GetMachineIP	Gets the dynamic IP address of the remote machine.
59	GetDepartName(**deprecated**)	Gets the department name from the machine.
60	SetDepartName(**deprecated**)	Sets the department name to the machine.
61	StartEventCapture	Starts capturing events from the machines.
62	StopEventCapture	Suspends capturing events from the machines.
63	ConnectP2p	P2P Connection establishment.
4 Event Handlers		
1	OnReceiveEventXML	Called back (by XML) when receiving an event from a machine.

2. Properties

2.1 CommPort (deprecated)

The port number of serial communication. This property is set before call of *OpenCommPort*.

Type (C++)

```
long    CommPort
```

Type (C#)

```
int     CommPort
```

Initial Value

1(COM1)

Description

See *OpenCommPort*.

2.2 Baudrate (deprecated)

The baudrate of serial communication. This property is set before call of *OpenCommPort*.

Type (C++)

```
long    Baudrate
```

Type (C#)

```
int     Baudrate
```

Initial Value

115200

Description

See *OpenCommPort*.

2.3 ReadMark

The flag which represents whether obtained logs are marked as read.

Type (C++)

```
BOOL    ReadMark
```

Type (C#)

```
bool    ReadMark
```

Initial Value

TRUE

Description

See log getting functions.

3. Methods

3.1 SetMachineType

This function is used to set target machine type.

Syntax (C++)

```
BOOL SetMachineType(
    BSTR lpszMachineType);
```

Syntax (C#)

```
bool SetMachineType(
    string lpszMachineType);
```

Parameters

lpszMachineType : String of machine type.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

It is normally needed for legacy devices only, because new devices inform its type while connecting.
String of a machine type is confidential, *Smackbio* makes customer-specific document for this information.
If there is no customer-specific document, do not care about machine type, DO NOT CALL THIS FUNCTION.

3.2 DotNET

Informs that client development environment is .NET.

Syntax (C++)

```
void DotNET();
```

Syntax (C#)

```
void DotNET();
```

Parameters

None.

Return Values

None.

3.3 GetEnrollData

Gets an enrollment data from the machine.

Syntax (C++)

```
BOOL GetEnrollData(
    long        dwMachineNumber,
    long        dwEnrollNumber,
    long        dwEMachineNumber,
    long        dwBackupNumber,
    long*       dwMachinePrivilege,
    VARIANT*    dwEnrollData,
    long*       dwPassWord);
```

Parameters

dwMachineNumber : Target machine ID.
dwEnrollNumber : User ID.
dwEMachineNumber : Not used, set as target machine ID.
dwBackupNumber : Backup number (originally, it means finger number) of the data to get.
Below are values of the backup numbers.

Value	Meaning	Future
0~9	Fingerprint data for finger number from No.0 to No.9	
10	Password data (beginning with 0 is not allowed.)	
11	Card data	
14	User Timezone	deprecated, supported for SB2900, SB2960 (SB3600) only
15	Password data (beginning with 0 is allowed.)	deprecated, supported for SB2900, SB2960 (SB3600) only
16	User department	deprecated, supported for SB2960 (SB3600) only
17	Face data	Supported for F500

dwMachinePrivilege : [OUT] User level.
Below are values of user levels.

Value	Level	Meaning
0	Employee	User
1	Administrator	Supervisor
2	Enrollment manager	Manager who can enroll and setup, cannot manage supervisor & other managers.
3	Setup manager	Setup only

dwEnrollData : [OUT] The repository for fingerprint data. Its size is 1404+12 Bytes.
dwPassWord : [OUT] The repository for password or card value.
The password consists of four digits.
If *dwBackupNumber* == 14, 16, then *dwPassWord* has special pattern per product. Please refer sample source for more detail.
If *dwBackupNumber* == *15*, then the value of *dwPassWord* was encoded by special rule to preserve beginning zeros. So you should decode to get proper value. Please refer sample source for more detail.

Return Values

TRUE (nonzero): success.

FALSE (zero): failure.

Remarks

When *dwBackupNumber* is in 0~9, then value of *dwPassWord* is meaningless, only the value of *dwEnrollData* is meaningful.
Of course, reverse case is also true.
.NET client software must use *GetEnrollData1*

3.4 GetEnrollData1(for .NET users)

It is same as ***GetEnrollData*** except for the type of the parameter ***dwEnrollData***.

Syntax (C++)

```
BOOL GetEnrollData1(
    long        dwMachineNumber,
    long        dwEnrollNumber,
    long        dwBackupNumber,
    long*       dwMachinePrivilege,
    long*       dwEnrollData,
    long*       dwPassWord);
```

Syntax (C#)

```
bool GetEnrollData1(
    int         dwMachineNumber,
    int         dwEnrollNumber,
    int         dwBackupNumber,
    ref int     dwMachinePrivilege,
    ref int     dwEnrollData,
    ref int     dwPassWord);
```

3.5 SetEnrollData

Sets an enrollment data to the machine.

Syntax (C++)

```
BOOL SetEnrollData(
    long        dwMachineNumber,
    long        dwEnrollNumber,
    long        dwEMachineNumber,
    long        dwBackupNumber,
    long        dwMachinePrivilege,
    VARIANT*    dwEnrollData,
    long        dwPassWord);
```

Parameters

dwMachineNumber	: Target machine ID.
dwEnrollNumber	: User ID.
dwEMachineNumber	: Not used, set as target machine ID.
dwBackupNumber	: Backup number of the data to set. See *GetEnrollData*. If backup number is in 0~9, then double checking of fingerprint data will be required. If backup number is in 20~29, then it is same as in 0~9, but double checking of fingerprint data will not be required.
dwMachinePrivilege	: User level. See *GetEnrollData*.
dwEnrollData	: The repository for fingerprint data. Its size is 1404+12 Bytes. It must be valid fingerprint data, which was saved through *GetEnrollData*.
dwPassWord	: The repository for password or card value.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

When *dwBackupNumber* is in 0~9 (20~29), then value of *dwPassWord* is meaningless, only the value of *dwEnrollData* is meaningful.
Of course, reverse case is also true.
.NET client software must use *SetEnrollData1*

3.6 SetEnrollData1(for .NET users)

It is same as ***SetEnrollData*** except for the type of the parameter ***dwEnrollData***.

Syntax (C++)

```
BOOL SetEnrollData1(
    long        dwMachineNumber,
    long        dwEnrollNumber,
    long        dwBackupNumber,
    long        dwMachinePrivilege,
    long*       dwEnrollData,
    long        dwPassWord);
```

Syntax (C++)

```
bool SetEnrollData1(
    int         dwMachineNumber,
    int         dwEnrollNumber,
    int         dwBackupNumber,
    int         dwMachinePrivilege,
    ref int     dwEnrollData,
    int         dwPassWord);
```

3.7 DeleteEnrollData

Removes an enrollment data from the machine.

Syntax (C++)

```
BOOL DeleteEnrollData(
    long        dwMachineNumber,
    long        dwEnrollNumber,
    long        dwEMachineNumber,
    long        dwBackupNumber);
```

Syntax (C#)

```
bool DeleteEnrollData(
    int         dwMachineNumber,
    int         dwEnrollNumber,
    int         dwEMachineNumber,
    int         dwBackupNumber);
```

Parameters

dwMachineNumber : Target machine ID.
dwEnrollNumber : User ID.
dwEMachineNumber : Not used, set as target machine ID.
dwBackupNumber : Backup number of the data to delete. Below are values of the backup numbers.

Value	Meaning
0~9	Fingerprint data for finger number from No.0 to No.9
10	Password data
11	Card data
12	Delete all fingerprint , password, card data
13	Delete all fingerprint data only.
17	Face data

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.8 ReadSuperLogData

Reads the newly recorded management logs from the machine and stores it into the PC memory.

Syntax (C++)

```
BOOL ReadSuperLogData(
    long dwMachineNumber);
```

Syntax (C#)

```
bool ReadSuperLogData(
    long dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

Use *GetSuperLogData* to parse stored logs.
If *ReadMark* property is *TRUE*, then stored logs are marked as read at the device.

3.9 GetSuperLogData

Parses a management log from the PC memory.

Syntax (C++)

```
BOOL GetSuperLogData(
    long        dwMachineNumber,
    long*       dwTMachineNumber,
    long*       dwSEnrollNumber,
    long*       dwSMachineNumber,
    long*       dwGEnrollNumber,
    long*       dwGMachineNumber,
    long*       dwManipulation,
    long*       dwBackupNumber,
    long*       dwYear,
    long*       dwMonth,
    long*       dwDay,
    long*       dwHour,
    long*       dwMinute,
    long*       dwSecond);
```

Syntax (C#)

```
bool GetSuperLogData(
    int         dwMachineNumber,
    ref int     dwTMachineNumber,
    ref int     dwSEnrollNumber,
    ref int     dwSMachineNumber,
    ref int     dwGEnrollNumber,
    ref int     dwGMachineNumber,
    ref int     dwManipulation,
    ref int     dwBackupNumber,
    ref int     dwYear,
    ref int     dwMonth,
    ref int     dwDay,
    ref int     dwHour,
    ref int     dwMinute,
    ref int     dwSecond);
```

Parameters

dwMachineNumber :Target machine ID.
dwTMachineNumber :[OUT] Not used, it is same as the target machine ID.

dwSEnrollNumber :[OUT] The ID value of the manager, who performed the action.
If this value is 0, then it means that no manager enrolled.
dwSMachineNumber :[OUT] Not used, it is same as the target machine ID.
dwGEnrollNumber :[OUT] The ID value of the user, who received the action.
If this value is 0, then it means that the action is related with the device itself.
dwGMachineNumber :[OUT] Not used, it is same as the target machine ID.
dwManipulation :[OUT] Value of the action.
Below are the values of the action.

Value	Meaning
3	A user has been enrolled.
4	A manager has been enrolled.
5	A fingerprint data has been deleted.
6	A password data has been deleted.
7	A card data has been deleted.
8	All log data has been deleted.
9	A Setting has been changed.
10	The date/time has been changed.
11	A log setting has been changed.
12	A communication setting has been changed.
13	The time zone setting been changed.
14	A face data has been deleted.

dwBackupNumber :[OUT] Backup number, which the action was related with.
Below are values of the backup number.

Value	Meaning
0~9	Fingerprint data for finger number from No.0 to No.9
10	Password data
13	Card data
14	Face data

dwYear, dwMonth, dwDay, dwHour, dwMinute, dwSecond
: [OUT] The time, when the action was occurred.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure, it means that there are no more logs to parse.

Remarks

This function parses a management log from the PC memory, which filled by *ReadSuperLogData*.

3.10 ReadGeneralLogData

Reads the newly recorded time logs from the machine and stores it into the PC memory.

Syntax (C++)

```
BOOL ReadGeneralLogData(
  long    dwMachineNumber);
```

Syntax (C#)

```
bool ReadGeneralLogData(
  int     dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

Use *GetGeneralLogData* to parse stored logs.
If *ReadMark* property is *TRUE*, then stored logs are marked as read at the device.

3.11 GetGeneralLogData

Parses a time log from the PC memory.

Syntax (C++)

```
BOOL GetGeneralLogData(
  long    dwMachineNumber,
  long*   dwTMachineNumber,
  long*   dwEnrollNumber,
  long*   dwEMachineNumber,
  long*   dwVerifyMode,
  long*   dwYear,
  long*   dwMonth,
  long*   dwDay,
  long*   dwHour,
  long*   dwMinute,
  long*   dwSecond);
```

Syntax (C#)

```
bool GetGeneralLogData(
  int           dwMachineNumber,
  ref int       dwTMachineNumber,
  ref int       dwEnrollNumber,
  ref int       dwEMachineNumber,
  ref int       dwVerifyMode,
  ref int       dwYear,
  ref int       dwMonth,
  ref int       dwDay,
  ref int       dwHour,
  ref int       dwMinute,
  ref int       dwSecond);
```

Parameters

dwMachineNumber : Target machine ID.
dwTMachineNumber :[OUT] Not used, it is same as the target machine ID. For some machine types it has special meaning.

Machine Type	Meaning
SB3600	Logged photo index, if -1, then it is invalid.

EC500	Logged job code.

dwEnrollNumber :[OUT] The ID value of the user, who clocked in/out.
dwEMachineNumber :[OUT] Not used, it is same as the target machine ID.
dwVerifyMode :[OUT] Clocking mode.
Below values are available.

Little Endian LSB BYTE 0

Value	Meaning
1	Fingerprint-based
2	Password-based
3	Card-based
4	Fingerprint + Card
5	Fingerprint + Password
6	Card + Password
7	Fingerprint + Card + Password
51	In : Fingerprint-based
52	In : Password-based
53	In : Card-based
101	Out : Fingerprint-based
102	Out : Password-based
103	Out : Card-based
151	Extra: Fingerprint-based
152	Extra: Password-based
153	Extra: Card-based
10	(access control) Hand lock
11	(access control) Program lock
12	(access control) Program open
13	(access control) Program close
14	(access control) Auto recover
20	(access control) Lock over
21	(access control) Illegal open
22	(access control) Duress alarm
23	(access control) Tampering detected
30	Face-based
31	Face + Card
32	Face + Password
33	Face + Card + Password
34	Face + Fingerprint

Little Endian LSB BYTE 1 – attendance status

0 duty on
1 duty off
2 overtime on

3	overtime off
4	go in
5	go out

Little Endian LSB BYTE 2 – Anti-pass status

0	None
1	In
3	Out

dwYear, dwMonth, dwDay, dwHour, dwMinute, dwSecond
: [OUT] Clocking time.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure, it means that there are no more logs to parse.

Remarks

This function parses a time log from the PC memory, which filled by *ReadGeneralLogData*.

3.12 ReadAllSLogData

Reads all recorded management logs from the machine and stores it into the PC memory.
That is, this function ignores read mark.

Syntax (C++)

```
BOOL ReadAllSLogData(
    long dwMachineNumber);
```

Syntax (C#)

```
bool ReadAllSLogData(
    long dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

Use *GetALLSLogData* to parse stored logs.

3.13 GetAllSLogData

Parses a management log from the PC memory.

Syntax (C++)

```
BOOL GetAllSLogData(
    long        dwMachineNumber,
    long*       dwTMachineNumber,
    long*       dwSEnrollNumber,
    long*       dwSMachineNumber,
    long*       dwGEnrollNumber,
    long*       dwGMachineNumber,
    long*       dwManipulation,
    long*       dwBackupNumber,
    long*       dwYear,
    long*       dwMonth,
    long*       dwDay,
    long*       dwHour,
    long*       dwMinute,
    long*       dwSecond);
```

Syntax (C#)

```
bool GetAllSLogData(
    int         dwMachineNumber,
    ref int     dwTMachineNumber,
    ref int     dwSEnrollNumber,
    ref int     dwSMachineNumber,
    ref int     dwGEnrollNumber,
    ref int     dwGMachineNumber,
    ref int     dwManipulation,
    ref int     dwBackupNumber,
    ref int     dwYear,
    ref int     dwMonth,
    ref int     dwDay,
    ref int     dwHour,
    ref int     dwMinute,
    ref int     dwSecond);
```

Remarks

This function is same as *GetSuperLogData*, except that it is used after calling of *ReadAllSLogData*.

3.14 ReadAllGLogData

Reads all recorded time logs from the machine and stores it into the PC memory.
That is, this function ignores read mark.

Syntax (C++)

```
BOOL ReadAllGLogData(
  long    dwMachineNumber);
```

Syntax (C#)

```
bool ReadAllGLogData(
  int     dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

Use *GetALLGLogData* to parse stored logs.

3.15 GetAllGLogData

Parses a time log from the PC memory.

Syntax (C++)

```
BOOL GetAllGLogData(
  long    dwMachineNumber,
  long*   dwTMachineNumber,
  long*   dwEnrollNumber,
  long*   dwEMachineNumber,
  long*   dwVerifyMode,
  long*   dwYear,
  long*   dwMonth,
  long*   dwDay,
  long*   dwHour,
  long*   dwMinute,
  long*   dwSecond);
```

Syntax (C#)

```
bool GetAllGLogData(
  int            dwMachineNumber,
  ref int        dwTMachineNumber,
  ref int        dwEnrollNumber,
  ref int        dwEMachineNumber,
  ref int        dwVerifyMode,
  ref int        dwYear,
  ref int        dwMonth,
  ref int        dwDay,
  ref int        dwHour,
  ref int        dwMinute,
  ref int        dwSecond);
```

Remarks

This function is same as *GetGeneralLogData*, except that it is used after calling of *ReadAllGLogData*.

3.16 GetDeviceStatus

Gets a current status of the device.

Syntax(C++)

```
BOOL GetDeviceStatus(
  long          dwMachineNumber,
  long          dwStatus,
  long*         dwValue);
```

Syntax(C#)

```
bool GetDeviceStatus(
  int           dwMachineNumber,
  int           dwStatus,
  ref int       dwValue);
```

Parameters

dwMachineNumber : Target machine ID.

dwStatus : Type of the status info to get

Below are available values of this parameter.

Value	Meaning
1	The number of managers currently enrolled on the device
2	The number of users currently enrolled on the device
3	The number of fingerprints currently enrolled on the device
4	The number of passwords currently enrolled on the device
5	The number of management log currently recorded on the device
6	The number of time log currently recorded on the device
7	The number of cards currently enrolled on the device
8	Alarm status bit 0 – Opening time over bit 1 – Illegal open bit 2 – Duress verification bit 3 – Tampering detected
9	The number of faces currently enrolled on the device
10	The number of management log not read yet
11	The number of time log not read yet

dwValue : [OUT] The repository for the value of the status information

Return Values

TRUE (nonzero): success.

FALSE (zero): failure.

3.17 GetDeviceInfo

Gets a device parameter (setting) from the machine.

Syntax(C++)

```
BOOL GetDeviceInfo(
  long          dwMachineNumber,
  long          dwInfo,
  long*         dwValue);
```

Syntax(C#)

```
bool GetDeviceInfo(
  int           dwMachineNumber,
  int           dwInfo,
  ref int       dwValue);
```

Parameters

dwMachineNumber : Target machine ID.
dwInfo : Type of the device parameter to get
dwValue : [OUT] The repository for the value of the device parameter.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

Below are available values of the *dwInfo* paramter.

Value	Meaning	Range	
1	Maximum number of managers	0~10	
2	The machine ID.	1~255	
3	UI Language	0	English
		1	Chinese (simplified)
		2	Chinese (Traditional)
4	Auto power-off interval (minute)	0~9999	
5	Door opening time(second)	0~10	
6	Time log warning, if available space for time recording is less than this value, then an alarm will be occurred.	0~1000	
7	Management log warning, if available space of management recording is less than this value, then an alarm will be occurred.	0~100	
8	Re-verification interval (minute)	0~255	
9	Serial Baud rate	3	9,600 bps
		4	19,200 bps
		5	38,400 bps
		6	57,600 bps
		7	115,200 bps
10	Serial Parity check (no effect)	0	No making parity check
		1	Even check
		2	Odd check
11	Serial Stop bit (no effect)	0	Stop bit 1
		1	Stop bit 2

12	Date separator	0	'/'
		1	'-'
13	Verification mode	0	FP or Card or PWD
		1	FP & Card
		2	FP & PWD
		3	Card & PWD
		4	FP & Card & PWD
		6	FP
		7	Card
		9	ID&PWD
		11	FACE
		12	FACE & Card
		13	FACE & PWD
		14	Face & Card & PWD
		15	Face &FP
14	Mode of controlling the door	0	Unconditional Close
		1	Unconditional Open
		2	Auto
15	Door sensor type	0	None
		1	Always Open
		2	Always Close
16	Door opening timeout (minute)	0~255	
17	Anti-pass	0	Disable
		1	Enable
18	Auto sleep interval (minute)	0~9999	
19	Daylight offset	0~3	
20,21,22	(Reserved)		
23	Show Realtime Camera	0	None
		1	Enrolled Photo
		2	Realtime Camera
24	Use Fail Log	0	No Use
		1	Use

3.18 SetDeviceInfo

Sets a device parameter (setting) to the machine.

Syntax(C++)

```
BOOL SetDeviceInfo(
  long          dwMachineNumber,
  long          dwInfo,
  long          dwValue);
```

Syntax(C#)

```
bool SetDeviceInfo(
  int           dwMachineNumber,
  int           dwInfo,
  int           dwValue);
```

Parameters

dwMachineNumber : Target machine ID.
dwInfo : Type of the device parameter to set
dwValue : New value of the device parameter.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

For more information, see *GetDeviceInfo*.

3.19 EnableDevice

Enables (or disables) the machine for mutual access from the PC.

Syntax(C++)

```
BOOL EnableDevice(
  long          dwMachineNumber,
  long          bFlag);
```

Syntax(C#)

```
bool EnableDevice(
  int           dwMachineNumber,
  int           bFlag);
```

Parameters

dwMachineNumber : Target machine ID.
bflag : TRUE: enable the machine to allow user access.
FALSE: disable the machine for mutual access from the PC.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

It is recommended to disable the machine while accessing from the PC.
It is mandatory to disable the machine while transactional operation from the PC

3.20 EnableUser

Enables (or disables) a user.

Syntax(C++)

```
BOOL EnableUser(
  long   dwMachineNumber,
  long   dwEnrollNumber,
  long   dwEMachineNumber,
  long   dwBackupNumber,
  long   bFlag);
```

Syntax(C#)

```
bool EnableUser(
  int    dwMachineNumber,
  int    dwEnrollNumber,
  int    dwEMachineNumber,
  int    dwBackupNumber,
  int    bFlag);
```

Parameters

dwMachineNumber : Target machine ID.
dwEnrollNumber : User ID
dwEMachineNumber : Not used, set as target machine ID.
dwBackupNumber : Backup number, see *dwBackupNumber* of *DeleteEnrollData*.
bflag : TRUE: enable the user.
FALSE: disable the user.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

Most products do not support individual enabling/disabling per credential (indicated by backup number), that is, enabling/disabling state is unique per user.
This means that *dwBackupNumber* is really meaningless.

3.21 GetDeviceTime

Gets the current time of the machine.

Syntax(C++)

```
BOOL GetDeviceTime(
  long          dwMachineNumber,
  long*         dwYear,
  long*         dwMonth,
  long*         dwDay,
  long*         dwHour,
  long*         dwMinute,
  long*         dwSecond,
  long*         dwDayOfWeek);
```

Syntax(C#)

```
bool GetDeviceTime(
  int           dwMachineNumber,
  ref int       dwYear,
  ref int       dwMonth,
  ref int       dwDay,
  ref int       dwHour,
  ref int       dwMinute,
  ref int       dwSecond,
  ref int       dwDayOfWeek);
```

Parameters

dwMachineNumber : Target machine ID.
dwYear, dwMonth, dwDay, dwHour, dwMinute, dwSecond
: [OUT] Current time of the machine.
dwDayOfWeek : [OUT] The value of current weekday of the machine

Value	Meaning
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.22 SetDeviceTime

Sets the current time of the machine.

Syntax(C++)

```
BOOL SetDeviceTime(
  long          dwMachineNumber);
```

Syntax(C#)

```
bool SetDeviceTime(
  int           dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

Current PC time will be transferred.

3.23 PowerOnAllDevice

Powers on all the machines, which connected at the same RS-485 network.

Syntax(C++)

```
void     PowerOnAllDevice( void );
```

Syntax(C#)

```
void     PowerOnAllDevice( void );
```

Parameters

None.

Return Values

None.

Remarks

Some products maybe do not support RS-485 power on.

3.24 PowerOffDevice

Powers off a machine.

Syntax(C++)

```
BOOL PowerOffDevice(
  long    dwMachineNumber);
```

Syntax(C#)

```
bool PowerOffDevice(
  long    dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.25 ModifyPrivilege

Modifies the level of a user.

Syntax(C++)

```
BOOL ModifyPrivilege(
  long          dwMachineNumber,
  long          dwEnrollNumber,
  long          dwEMachineNumber,
  long          dwBackupNumber,
  long          dwMachinePrivilege);
```

Syntax(C#)

```
bool ModifyPrivilege(
  int           dwMachineNumber,
  int           dwEnrollNumber,
  int           dwEMachineNumber,
  int           dwBackupNumber,
  int           dwMachinePrivilege);
```

Parameters

dwMachineNumber : Target machine ID.
dwEnrollNumber : User ID
dwEMachineNumber : Not used, set as target machine ID.
dwBackupNumber : Backup number, see *dwBackupNumber* of *DeleteEnrollData*.
dwMachinePrivilege : User level, see *dwMachinePrivilege* of *GetEnrollData*.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

Most products do not support individual level per credential (indicated by backup number), that is, user level value is unique per user.
This means that *dwBackupNumber* is really meaningless.

3.26 ReadAllUserID

Reads the database summary of the machine and stores it into PC memory.

Syntax(C++)

```
BOOL ReadAllUserID(
  long    dwMachineNumber);
```

Syntax(C#)

```
bool ReadAllUserID(
  int     dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

Use *GetALLUserID* to parse stored summary.

3.27 GetAllUserID

Parses an item of the database summary from the PC memory.

Syntax(C++)

```
BOOL GetAllUserID(
  long          dwMachineNumber,
  long*         dwEnrollNumber,
  long*         dwEMachineNumber,
  long*         dwBackupNumber,
  long*         dwMachinePrivilege,
  long*         dwEnable);
```

Syntax(C#)

```
bool GetAllUserID(
  int           dwMachineNumber,
  ref int       dwEnrollNumber,
  ref int       dwEMachineNumber,
  ref int       dwBackupNumber,
  ref int       dwMachinePrivilege,
  ref int       dwEnable);
```

Parameters

dwMachineNumber : Target machine ID.
dwEnrollNumber :[OUT] User ID.
dwEMachineNumber :[OUT] Not used, it is same as the target machine ID.
dwBackupNumber :[OUT]Backup number
dwMachinePrivilege :[OUT]User level
dwEnable :[OUT]Value indicating user's enable/disable state, see below.

Little Endian LSB BYTE 0

Value	Meaning
1	Record of an attendance of this user has been permitted.
2	Record of an attendance of this user has been forbidden, that is, his/her attendance is not recorded in spite of his/her registration.

Little Endian LSB BYTE 1

Value	Meaning
0	Normal fingerprint.
1	Duress fingerprint(emit duress alarm on verification)

Return Values

TRUE (nonzero): success.
FALSE (zero): failure, it means that there are no more items to parse.

Remarks

This function parses an item of the database summary from the PC memory, which filled by *ReadAllUserID*.

3.28 GetSerialNumber

Gets the serial number of the machine.

Syntax(C++)

```
BOOL GetSerialNumber(
  long          dwMachineNumber,
  BSTR*         dwSerialNumber);
```

Syntax(C#)

```
bool GetSerialNumber(
  int           dwMachineNumber,
  ref string    dwSerialNumber);
```

Parameters

dwMachineNumber : Target machine ID.
dwSerialNumber : [OUT] Serial number of the machine, its type is string.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.29 GetBackupNumber

Gets the maximum allowable count of finger per user.

Syntax(C++)

```
long GetBackupNumber(
  long          dwMachineNumber);
```

Syntax(C++)

```
int GetBackupNumber(
  int           dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

Maximum allowable count of finger per user. Normally, its value is 3 or 10.
(Some products report other value, which has not above meaning.)

3.30 GetProductCode

Gets the product code of the machine.

Syntax(C++)

```
BOOL GetProductCode(
  long          dwMachineNumber,
  BSTR*         lpszProductCode);
```

Syntax(C#)

```
bool GetProductCode(
  int           dwMachineNumber,
  ref string    lpszProductCode);
```

Parameters

dwMachineNumber : Target machine ID.
lpszProductCode : [OUT] Product code of the machine, its type is string.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.31 ClearKeeperData

Clears all the (management & time) logs and user database.

Syntax(C++)

```
BOOL ClearKeeperData(
  long          dwMachineNumber);
```

Syntax(C#)

```
bool ClearKeeperData(
  int           dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.32 EmptyEnrollData

Clears user database.

Syntax(C++)

```
BOOL EmptyEnrollData(
  long          dwMachineNumber);
```

Syntax(C#)

```
bool EmptyEnrollData(
  int           dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.33 EmptyGeneralLogData

Clears all time logs.

Syntax(C++)

```
BOOL EmptyGeneralLogData(
  long          dwMachineNumber);
```

Syntax(C#)

```
bool EmptyGeneralLogData(
  int           dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.34 EmptySuperLogData

Clears all management logs.

Syntax(C++)

```
BOOL EmptySuperLogData(
  long          dwMachineNumber);
```

Syntax(C#)

```
bool EmptySuperLogData(
  int           dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.35 GetUserName

Gets the name of a user.

Syntax(C++)

```
BOOL GetUserName(
  long          DeviceKind,
  long          dwMachineNumber,
  long          dwEnrollNumber,
  long          dwEMachineNumber,
  VARIANT*      dwUserName);
```

Parameters

DeviceKind : Not used
dwMachineNumber : Target machine ID.
dwEnrollNumber : User ID
dwEmachineNumber : Not used, set as target machine ID.
dwUserName : [OUT] The repository for name string.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

.NET client software must use *GetUserName1*

3.36 GetUserName1(for .NET users)

It is same as ***GetUserName*** except for the type of the parameter ***dwUserName***.

Syntax(C++)

```
BOOL GetUserName1(
  long          dwMachineNumber,
  long          dwEnrollNumber,
  BSTR*         dwUserName);
```

Syntax(C#)

```
bool GetUserName1(
  int           dwMachineNumber,
  int           dwEnrollNumber,
  ref string    dwUserName);
```

3.37 SetUserName

Sets the name of a user.

Syntax(C++)

```
BOOL SetUserName(
  long          DeviceKind,
  long          dwMachineNumber,
  long          dwEnrollNumber,
  long          dwEMachineNumber,
  VARIANT*      dwUserName);
```

Parameters

DeviceKind	: Not used
dwMachineNumber	: Target machine ID.
dwEnrollNumber	: User ID
dwEmachineNumber	: Not used, set as target machine ID.
dwUserName	: The repository for name string.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

.NET client software must use *SetUserName1*

3.38 SetUserName1(for .NET users)

It is same as ***SetUserName*** except for the type of the parameter ***dwUserName***.

Syntax(C++)

```
BOOL SetUserName1(
  long          dwMachineNumber,
  long          dwEnrollNumber,
  BSTR*         dwUserName);
```

Syntax(C#)

```
bool SetUserName1(
  int           dwMachineNumber,
  int           dwEnrollNumber,
  ref string    dwUserName);
```

3.39 GetCompanyName

Gets the company name of the machine.

Syntax(C++)

```
BOOL GetCompanyName(
  long          dwMachineNumber,
  VARIANT*      dwCompanyName);
```

Parameters

dwMachineNumber	: Target machine ID.
dwCompanyName	: [OUT] The repository for company name string.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

.NET client software must use *GetCompanyName1*

3.40 GetCompanyName1(for .NET users)

It is same as ***GetCompanyName*** except for the type of the parameter ***dwCompanyName***.

Syntax(C++)

```
BOOL GetCompanyName1(
  long          dwMachineNumber,
  BSTR*         dwCompanyName);
```

Syntax(C#)

```
bool GetCompanyName1(
  int           dwMachineNumber,
  ref string    dwCompanyName);
```

3.41 SetCompanyName

Sets the company name of the machine.

Syntax(C++)

```
BOOL SetCompanyName(
  long          dwMachineNumber,
  long          vKind,
  VARIANT*      dwCompanyName);
```

Parameters

dwMachineNumber : Target machine ID.
vKind : The flag which determines whether display company name.
1 : Display, 0 : Do not display.
dwCompanyName : The repository for company name string.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

.NET client software must use *GetCompanyName1*

3.42 SetCompanyName1(for .NET users)

It is same as ***SetCompanyName*** except for the type of the parameter ***dwCompanyName***.

Syntax(C++)

```
BOOL SetCompanyName1(
  long          dwMachineNumber,
  long          vKind,
  BSTR*         dwCompanyName);
```

Syntax(C#)

```
bool SetCompanyName1(
  int           dwMachineNumber,
  int           vKind,
  ref string    dwCompanyName);
```

3.43 GetDoorStatus

Gets the door status of the machine.

Syntax(C++)

```
BOOL GetDoorStatus(
  long          dwMachineNumber,
  long*         dwValue);
```

Syntax(C#)

```
bool GetDoorStatus(
  int           dwMachineNumber,
  ref int       dwValue);
```

Parameters

dwMachineNumber : Target machine ID.
dwValue : [OUT]Current door status of the machine.
1 : Forced open
2 : Forced close
3 : Open
4 : Auto recover.
5 : Close
6 : Watching for Close
7 : Illegal Open.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.44 SetDoorStatus

Sets the door status of the machine.

Syntax(C++)

```
BOOL SetDoorStatus(
  long          dwMachineNumber,
  long          dwValue);
```

Syntax(C#)

```
bool SetDoorStatus(
  int           dwMachineNumber,
  int           dwValue);
```

Parameters

dwMachineNumber : Target machine ID.
dwValue : New door status of the machine.
1 : Forced open
2 : Forced close
3 : Program open
4 : Auto recover.
5 : Reboot
6 : Cancel Warning.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.45 GetBellTime

Gets the ringing time of the machine.

Syntax(C++)

```
BOOL GetBellTime(
  long          dwMachineNumber,
  long*         dwValue,
  long*         dwBellInfo);
```

Syntax(C#)

```
bool GetBellTime(
  int           dwMachineNumber,
  ref int       dwValue,
  ref int       dwBellInfo);
```

Parameters

dwMachineNumber	: Target machine ID.
dwValue	: [OUT]Current ringing count. (1~12)
dwBellInfo	: [OUT] The repository for ringing times. 12 ringing times are allowed. 3 bytes (Validation flag, hour, minute) are required per a ringing time. So, *dwBellInfo* is a pointer to 36 bytes array. See sample source for more detail.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

The structure of *dwBellInfo* depends on machine type, so please refer proper sample source code per machine type.

3.46 SetBellTime

Sets the ringing time of the machine.

Syntax(C++)

```
BOOL SetBellTime(
  long          dwMachineNumber,
  long          dwValue,
  long*         dwBellInfo);
```

Syntax(C#)

```
bool SetBellTime(
  int           dwMachineNumber,
  int           dwValue,
  ref int       dwBellInfo);
```

Parameters

dwMachineNumber	: Target machine ID.
dwValue	: New ringing count. (1~12)
dwBellInfo	:The repository for ringing times. 12 ringing times are allowed. 3 bytes (Validation flag, hour, minute) are required per a ringing time. So, *dwBellInfo* is a pointer to 36 bytes array. See sample source for more detail.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

The structure of *dwBellInfo* depends on machine type, so please refer proper sample source code per machine type.

3.47 ConnectSerial

Connects to the machine via the serial port (RS232, RS485) or USB.

Syntax(C++)

```
BOOL ConnectSerial(
  long          dwMachineNumber,
  long          dwCommPort,
  long          dwBaudRate);
```

Syntax(C#)

```
bool ConnectSerial(
  int           dwMachineNumber,
  int           dwCommPort,
  int           dwBaudRate);
```

Parameters

dwMachineNumber : Target machine ID.

dwCommPort : Communication port of the PC.

Value	Meaning
0	USB
1	COM1
2	COM2
3	COM3
4	COM4

dwBaudRate : Baud rate of the communication port.

Value
9600
19200
38400
57600
115200

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.48 ConnectTcpip

Connects to the machine via the Ethernet (TCP/IP).

Syntax(C++)

```
BOOL ConnectTcpip(
  long          dwMachineNumber,
  BSTR*         lpszIPAddress,
  long          dwPortNumber,
  long          dwPassWord);
```

Syntax(C#)

```
bool ConnectTcpip(
  int           dwMachineNumber,
  ref string    lpszIPAddress,
```

```
  int           dwPortNumber,
  int           dwPassWord);
```

Parameters

dwMachineNumber	: Target machine ID.
lpszIPAddress	: IP address of the marching, string format.
dwPortNumber	: TCP Port number of the machine. Default is 5005. This value must be 5005 for SB2880.
dwPassword	: Password for network communication.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.49 Disconnect

Disconnects the machine.

Syntax(C++)

```
Void          Disconnect(  void  )
```

Syntax(C#)

```
Void          Disconnect(  void  )
```

Parameters

None.

Return Values

None.

3.50 SetIPAddress (deprecated)

Set IP address of target machine. This function is called before call of *OpenCommPort*.

Syntax(C++)

```
BOOL SetIPAddress(
  BSTR*         IPAddress,
  long          dwPortNumber,
  long          dwPassword);
```

Syntax(C#)

```
bool SetIPAddress(
  ref string    IPAddress,
  int           dwPortNumber,
  int           dwPassword);
```

Parameters

IPAddress	: IP address of the marching, string format.
dwPortNumber	: TCP Port number of the machine. Default is 5005. This value must be 5005 for SB2880.
dwPassword	: Password for network communication.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.51 OpenCommPort (deprecated)

Connects to the machine via the serial port (RS232, RS485) or USB or TCP/IP.
If client software changes *CommPort* or *Baudrate* property, then communication channel will be serial port.
If client software calls *SetIPAddress*, then communication channel will be TCP/IP.

Syntax(C++)

```
BOOL OpenCommPort(
        long        dwMachineNumber);
```

Syntax(C#)

```
bool OpenCommPort(
        int         dwMachineNumber);
```

Parameters

dwMachineNumber : Target machine ID.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.52 CloseCommPort (deprecated)

This is an alias of *Disconnect*.

3.53 GetLastError

Gets the last error code.

Syntax(C++)

```
void GetLastError(
        long*       dwErrorCode);
```

Syntax(C#)

```
void GetLastError(
        ref long    dwErrorCode);
```

Parameters

dwErrorCode : [OUT] The repository for error code

Value	Meaning
0	Success.
1	Cannot open serial port.
2	Error on transmitting data.
3	Error on receiving data
4	Invalid parameter
5	Failure
6	No more data to parse.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.54 GetDeviceLongInfo(deprecated)

Gets a device parameter (big, its length > 4bytes) from the machine.

Syntax(C++)

```
BOOL GetDeviceLongInfo(
  long          dwMachineNumber,
  long          dwInfo,
  long*         dwValue);
```

Syntax(C#)

```
bool GetDeviceLongInfo(
  int           dwMachineNumber,
  int           dwInfo,
  ref int       dwValue);
```

Parameters

dwMachineNumber : Target machine ID.
dwInfo : Type of the device parameter to get
dwValue : [OUT] The repository for the value of the device parameter. The data size is variable depending on *dwInfo*.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

The data acquired by this function are the same as those shown on the "Settings" menu on the terminal excepting the date separator.

Below are values of the setup information.

Value	Meaning	Data size	Structure
1	Daylight start time	16bytes	array of long-type values(month, day, hour, minute)
2	Daylight end time	16bytes	array of long-type values(month, day, hour, minute)
3	Time zone	6400bytes	(50 set) * (8 day/set) * (4 long-type value/day) Day : Sunday~Saturday,Holiday Value : Start hour, Start minute, End hour, End minute
4	Verification mode time zone	200bytes	(10 set) * (5 long-type value/set) Value : Mode, Start hour, Start minute, End hour, End minute Mode : 0 FP or Card or PWD 1 FP & Card 2 FP & PWD 3 Card & PWD 4 FP & Card & PWD 5 System Default VM 6 FP 7 Card 9 ID&PWD 11 FACE 12 FACE & Card 13 FACE & PWD 14 FACE & Card & PWD 15 FACE & FP

5	Attendance status time zone	200bytes	(10 set) * (5 long-type **value**/set) **value** : **Mode**, Start hour, Start minute, End hour, End minute **Mode** : 0 duty on 1 duty off 2 overtime on 3 overtime off 4 go in 5 go out
6	Holiday	3072bytes	(256 set) * (3 long-type **value**/set) **value** : month, day, **period** 0 <= **period** <= 7 ※) If any one of month, day and period is zero, the set is invalid.

3.55 SetDeviceLongInfo(deprecated)

Sets a device parameter (big, its length > 4bytes) to the machine.

Syntax(C++)

```
BOOL SetDeviceInfo(
        long        dwMachineNumber,
        long        dwInfo,
        long*       dwValue);
```

Syntax(C#)

```
bool SetDeviceInfo(
        int         dwMachineNumber,
        int         dwInfo,
        ref int     dwValue);
```

Parameters

dwMachineNumber : Target machine ID.
dwInfo : Type of the device parameter to set, the meaning of this parameter is same as in *GetDeviceLongInfo*.
dwValue : The repository for the value of the device parameter, the meaning of this parameter is same as in *GetDeviceLongInfo*.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.56 ModifyDuressFP(deprecated)

Modifies duress status of a fingerprint.

Syntax(C++)

```
BOOL ModifyDuressFP (
  long          dwMachineNumber,
  long          dwEnrollNumber,
  long          dwBackupNumber,
  long          dwDuressSetting);
```

Syntax(C#)

```
bool ModifyDuressFP (
  int           dwMachineNumber,
  int           dwEnrollNumber,
  int           dwBackupNumber,
  int           dwDuressSetting);
```

Parameters

dwMachineNumber : Target machine ID.
dwEnrollNumber : User ID.
dwBackupNumber : fingerprint number, 0~9
dwDuressSetting : 0 – Normal fingerprint, 1 – duress fingerprint (alarm on verification).

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

3.57 SetDeviceTime1

Sets the time of the machine.

Syntax(C++)

```
BOOL SetDeviceTime(
  long          dwMachineNumber,
  long          dwYear,
  long          dwMonth,
  long          dwDay,
  long          dwHour,
  long          dwMinute,
   long         dwSecond);
```

Syntax(C#)

```
bool SetDeviceTime(
  int           dwMachineNumber,
  int           dwYear,
  int           dwMonth,
  int           dwDay,
  int           dwHour,
  int           dwMinute,
  int           dwSecond);
```

Parameters

dwMachineNumber : Target machine ID.
dwYear, dwMonth, dwDay, dwHour, dwMinute, dwSecond
: Time to be set.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

The specified time will be set to the machine. The applicable time range: 2000/01/01 00:00:00 ~ 2099/12/31 23:59:59.

4. Next generation of SBXPC with XML

To support (almost) all products with same OCX, new specs will be implemented through *GeneralOperationXML*.
More spec modification is forbidden.

4.1 GeneralOperationXML

Send XML request & receive XML response.

Syntax(C++)

```
BOOL    GeneralOperationXML( BSTR FAR* lpszReqNResXML );
```

Syntax(C#)

```
bool    GeneralOperationXML( ref string lpszReqNResXML );
```

Parameters

`lpszReqNResXML` : [IN] XML Request.
[OUT] XML Response.

Return Values

TRUE (nonzero): success.
FALSE (zero): failure.

Remarks

The syntax of XML request & XML response depend on machine type & operation.
Best reference is only sample source, please see proper sample source.
To make XML & parse XML, following helper functions are supported.

Syntax(MIDL)

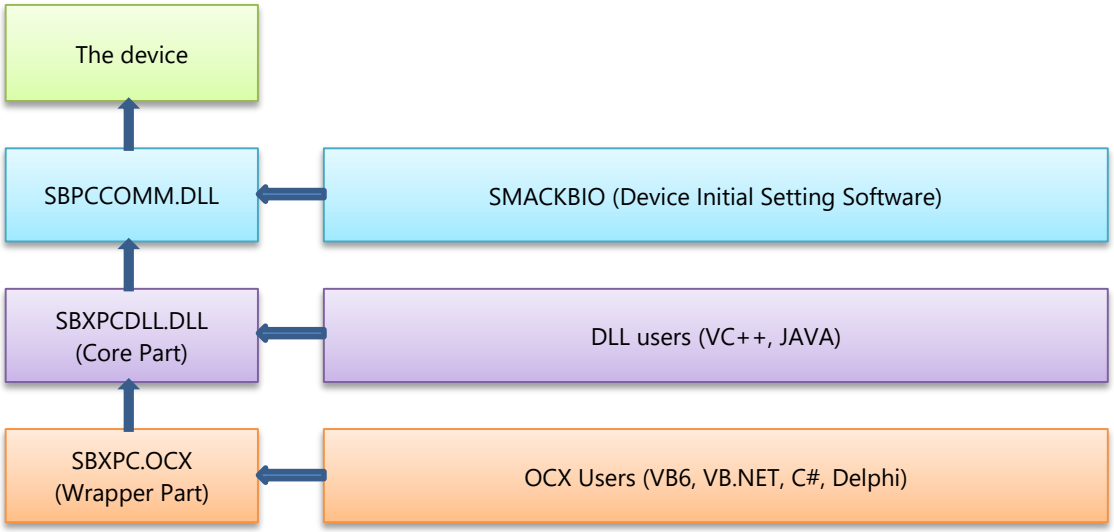
```
 long XML_ParseInt(BSTR* lpszXML, BSTR lpszTagname);
 long XML_ParseLong(BSTR* lpszXML, BSTR lpszTagname);
 boolean XML_ParseBoolean(BSTR* lpszXML, BSTR lpszTagname);
 boolean XML_ParseString(BSTR* lpszXML, BSTR lpszTagname, BSTR* lpszValue);
 boolean XML_ParseBinaryByte(BSTR* lpszXML, BSTR lpszTagname, BYTE* pData, long dwLen);
 boolean XML_ParseBinaryWord(BSTR* lpszXML, BSTR lpszTagname, WORD* pData, long dwLen);
 boolean XML_ParseBinaryLong(BSTR* lpszXML, BSTR lpszTagname, long* pData, long dwLen);
 boolean XML_ParseBinaryUnicode(BSTR* lpszXML, LPCTSTR lpszTagname, BSTR* pData, long dwLen);

 boolean XML_AddInt(BSTR* lpszXML, BSTR lpszTagname, int nValue);
 boolean XML_AddLong(BSTR* lpszXML, BSTR lpszTagname, long dwValue);
 boolean XML_AddBoolean(BSTR* lpszXML, BSTR lpszTagname, boolean bValue);
 boolean XML_AddString(BSTR* lpszXML, BSTR lpszTagname, BSTR lpszValue);
 boolean XML_AddBinaryByte(BSTR* lpszXML, BSTR lpszTagname, BYTE* dwData, long dwLen);
 boolean XML_AddBinaryWord(BSTR* lpszXML, BSTR lpszTagname, WORD* dwData, long dwLen);
 boolean XML_AddBinaryLong(BSTR* lpszXML, BSTR lpszTagname, long* dwData, long dwLen);
 boolean XML_AddBinaryUnicode(BSTR* lpszXML, BSTR lpszTagname, BSTR* lpszData);
 boolean  XML_AddBinaryGlyph(BSTR*  lpszXML,  BSTR  lpszMessage,  long  width,  long  height,  BSTR
lpszFontDescriptor);
```

If an operation is supported through both raw & XML interface, then the XML interface is preferred.

5. DLL mode support

To help native C++ developers & OCX-customers, SBXPC is separated into wrapper part (OCX format) and core part (DLL format).
Below is its diagram.



For OCX users, ***SBXPC.OCX***, ***SBXPCDLL.DLL***, ***SBPCCOMM.DLL*** should be installed in same folder.
For DLL users, ***SBXPCDLL.DLL***, ***SBPCCOMM.DLL*** should be installed in same folder.

SBXPC.OCX is deployed as source; this will help C++ developers & OCX-customers.
SMACKBIO guarantees binary-level compatibility for all versions of ***SBXPC.OCX***.

SBXPCDLL_API.h describes C syntax of ***SBXPCDLL.DLL*** API.
All functions have same syntax as OCX interface, except for leading underscore.
SBXPCDLL.lib is import library for ***SBXPCDLL.DLL***.

Below are notes for C++ developers.

- If a BSTR* argument has ***__needfree*** specifier, C++ application SHOULD free return BSTR value using ***SysFreeString***.
- If a function has a VARIANT argument, C++ application SHOULD not use it.
- If a function has ***_EXT*** suffix, it is for old SB3000 series, DO NOT TRY TO CALL IT.

6. Multi-device safe, Multi-thread safe support

One hosting program can make several OCX instances.
Several OCX instances will be distinguished by machine number, that is, different OCX instances must connect to the different machine ID.
This means that SBXPC is multi-device safe.

Moreover each OCX instances can be controlled concurrently.
This means that SBXPC is multi-thread safe.

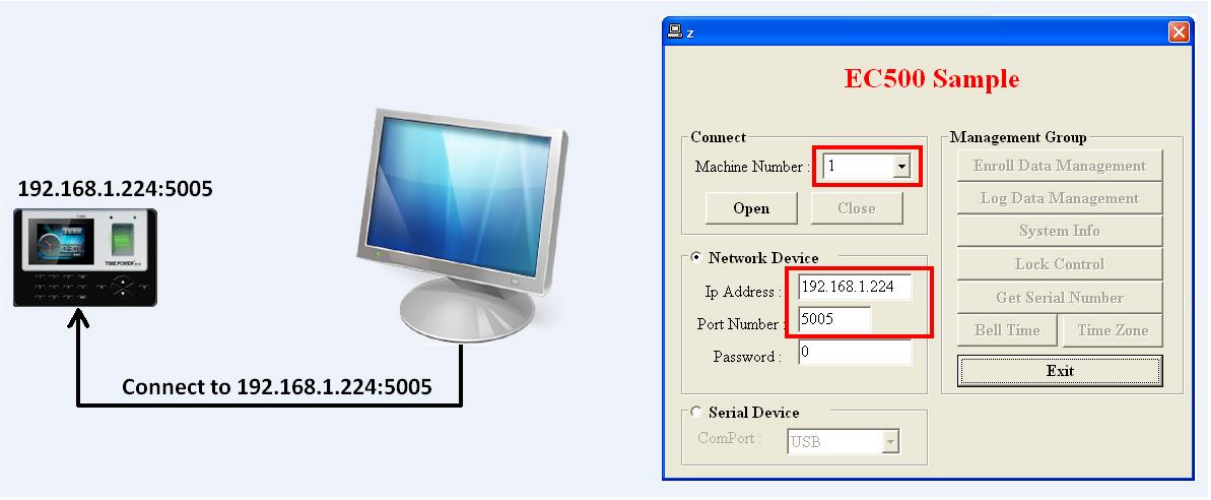
7. "TCP active connection" mode support

There is a condition where the device cannot accept TCP/IP connection from the PC.
In this condition SBXPC accepts TCP/IP connection from the device.
(This function is supported in EC500.)

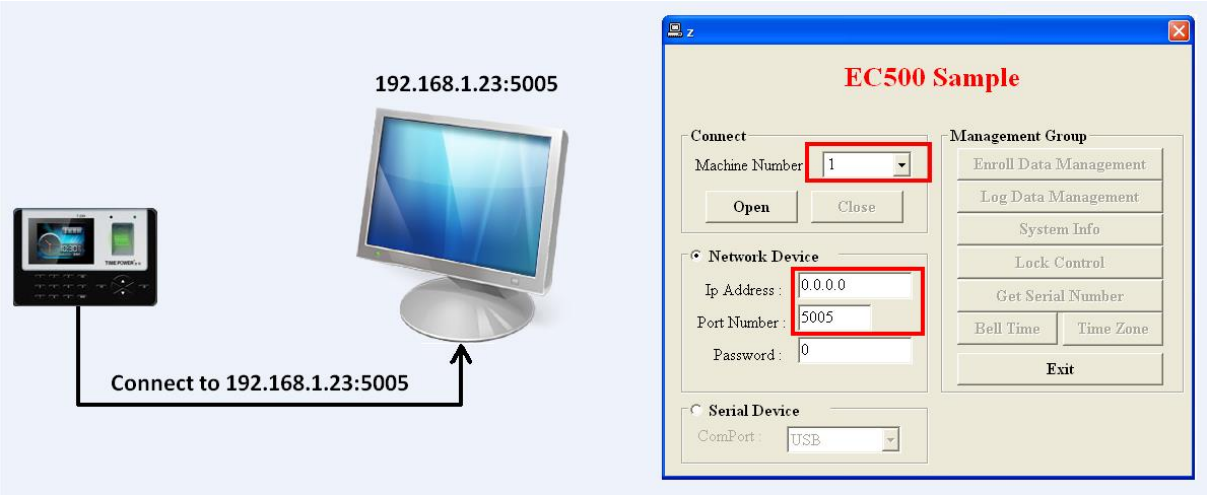
To use "TCP active connection", the hosting program calls connection-establishment functions with target device's IP = 0.0.0.0.
Below is detailed description.

Changes from the previous version are as follows.

- A new submenu item **"TCP connect mode"** has been created in **"Set IP"** menu.
- When this parameter is set to **"Passive"** (default value), EC500 time clock works just same as before. (PC starts connection.)



- When this parameter is set to **"Active"**, user should set IP address parameter to **"0.0.0.0"** while calling "**ConnectTcpip**" function. In this case, time clock starts TCP connection.



NOTE: As time clock tries to connect to PC in every 10s, it may take about 5~10s after you press "Open" button in the above sample.

Setting value of "TCP connect mode" option	**Required parameters in time clock**	**Required parameters in "ConnectTcpip" function**
Passive	Time clock's TCP port number	Time clock's IP address and TCP port number
Active	PC's IP address and TCP port number	PC's port number (Set IP address to 0.0.0.0)

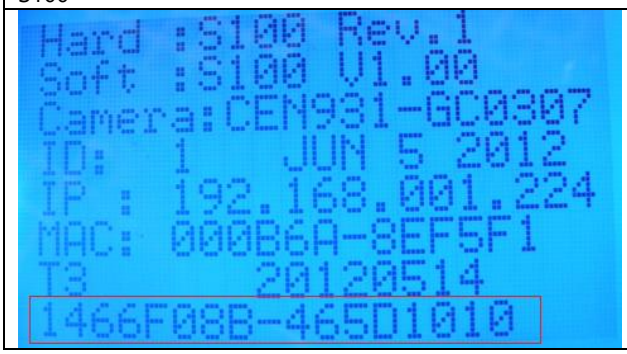
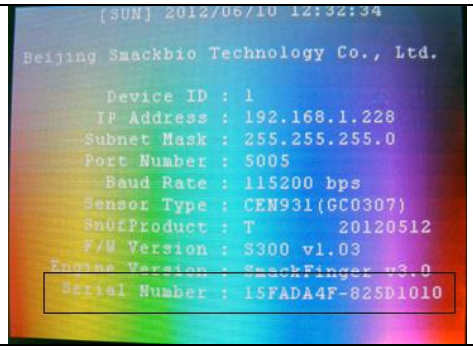
8. P2P (Peer To Peer) support

Before, to connect to a device through TCP/IP, the user must know the device's IP address.
Therefore controlling of the device through the Internet was impossible, because the device cannot have public IP address.
P2P mode resolves such problem.

Premise: There is a global server with public IP, the server supports global bridge service. Theoretically, it is need only one server in the Earth.

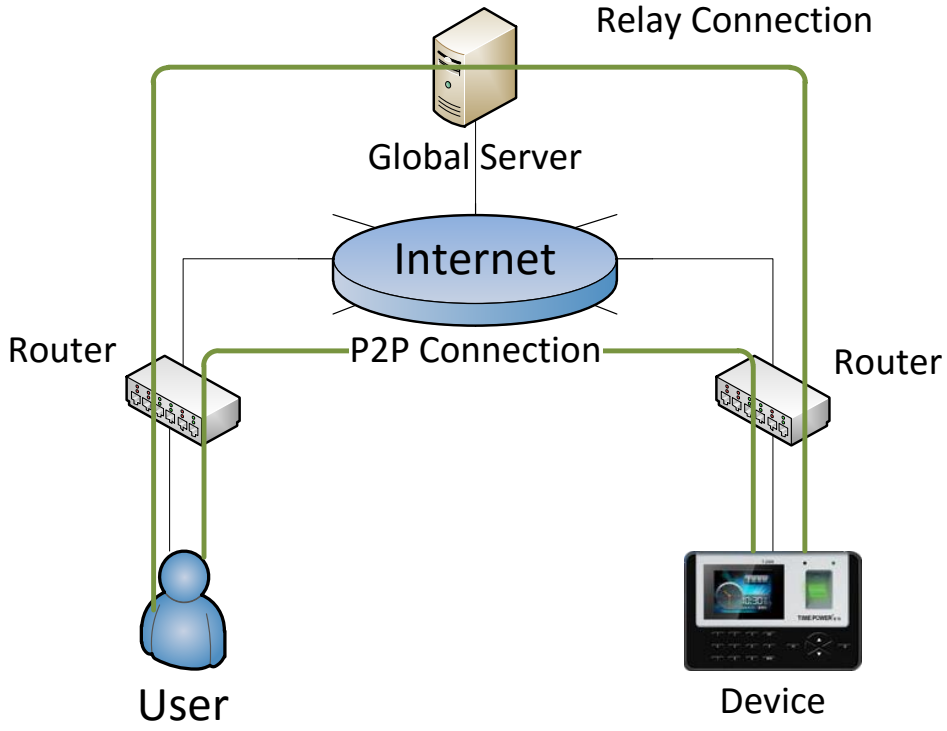
User's activity:

- No change for management software excepting connection-establishment part.
- When connecting to the device, enter [Server's public IP & TCP port] and [The device's global unique ID, 8bytes]. The user can see the device's global unique ID at the diagnostics screen which appeared when press ESC (or MENU) key while booting.

S100	S300
	

- The device is configured as using DHCP. The device is connected to the Internet. The device has menu item to enter [Server's public IP & TCP port].

Technical Insight:



The server makes best effective connection (P2P connection) between the user PC & the device. If the P2P connection is once established, the server does not interfere.
For example, if both the user PC & the device are in same LAN, the P2P connection will be established without the Internet.
If P2P connection was not established for some reason (firewall, NAT configuration), the server relays between the user PC & the device, this connection is relay connection.
Maybe, relay connection's speed is lower than P2P connection.

New OCX interface is as below.

8.1 ConnectP2p

Establish P2P connection.

Syntax(C++)

```
long ConnectP2p(
    BSTR* lpszMachineID,
    BSTR* lpszServerIPAddress,
    long dwServerPortNumber,
    long dwPassWord,
    long* pnError);
```

Syntax(C#)

```
int ConnectP2p(
    ref string lpszMachineID,
    ref string lpszServerIPAddress,
    int dwServerPortNumber,
    int dwPassWord,
    ref int pnError);
```

Parameters

`lpszMachineID` : [IN] The device's global unique ID, 8bytes hexadecimal string. For example, "15FADA4F825D1010"
`lpszServerIPAddress` : [IN] The server's IP address.
`dwServerPortNumber` : [IN] The server's TCP port number. Default value is 4000.
`dwPassWord` : [IN] The device's communication password.
`pnError` : [OUT] Extended error value.

Value	Meaning
0	The connection is P2P connection if connection established. Unknown error if connection failed.
1	Connection failed, Cannot connect to the server
2	Connection failed, Device not found
3	Connection failed, Password mismatched
4	Connection established, but relayed connection
5	Connection established, but direct local connection (the user PC & the device are in same LAN or the device has public IP.)

Return Values

Nonzero (0x10001 ~ 0x1FFFF): success. Connection established. The return value is "OCX Device ID".
0: failure. Connection failed

Remarks

The return value is "OCX Device ID". This mean that all subsequent OCX calls must use this returned "OCX Device ID" for "dwMachineNumber" parameter.

Appendix 1. Specific Machine Functionality

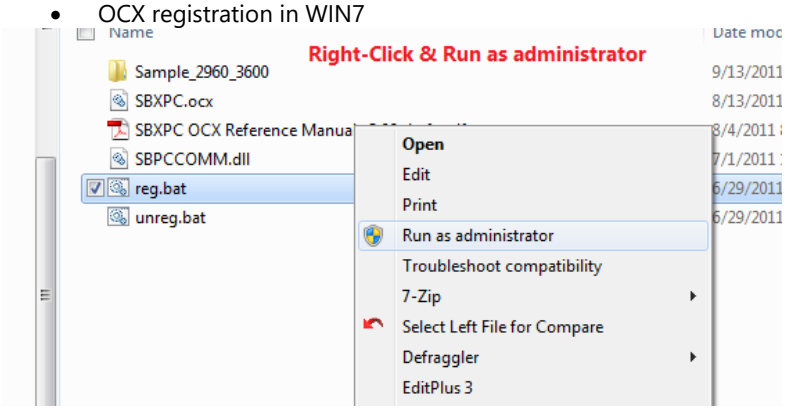
"+" refers to support, empty refers to NOT support

	Methods \ Machines	S100 (SB2900)	S300 (SB2960)	SB3600	SB2910 (EC500)
1	GetEnrollData/ GetEnrollData1	+	+	+	+
2	SetEnrollData/ SetEnrollData1	+	+	+	+
3	DeleteEnrollData	+	+	+	+
4	ReadSuperLogData	+	+	+	+
5	GetSuperLogData	+	+	+	+
6	ReadGeneralLogData	+	+	+	+
7	GetGeneralLogData	+	+	+	+
8	ReadAllSLogData	+	+	+	+
9	GetAllSLogData	+	+	+	+
10	ReadAllGLogData	+	+	+	+
11	GetAllGLogData	+	+	+	+
12	GetDeviceStatus	+	+	+	+
13	GetDeviceInfo	+	+	+	+
14	SetDeviceInfo	+	+	+	+
15	EnableDevice	+	+	+	+
16	EnableUser	+	+	+	+
17	GetDeviceTime	+	+	+	+
18	SetDeviceTime	+	+	+	+
19	PowerOnAllDevice				
20	PowerOffDevice	+	+	+	+
21	ModifyPrivilege	+	+	+	+
22	ReadAllUserID	+	+	+	+
23	GetAllUserID	+	+	+	+
24	GetSerialNumber	+	+	+	+
25	GetBackupNumber	+	+	+	+
26	GetProductCode	+	+	+	+
27	ClearKeeperData	+	+	+	+
28	EmptyEnrollData	+	+	+	+
29	EmptyGeneralLogData	+	+	+	+
30	EmptySuperLogData	+	+	+	+
31	GetUserName/ GetUserName1	+	+	+	+
32	SetUserName/ SetUserName1	+	+	+	+
33	GetCompanyName/ GetCompanyName1	+			
34	SetCompanyName/ SetCompanyName1	+			
35	GetDoorStatus	+	+	+	+
36	SetDoorStatus	+	+	+	+
37	GetBellTime	+	+	+	+
38	SetBellTime	+	+	+	+
39	ConnectSerial	+	+	+	+
40	ConnectTcpip	+	+	+	+
41	Disconnect	+	+	+	+
42	SetIPAddress	+	+	+	+
43	OpenCommPort	+	+	+	+

44	CloseCommPort	+	+	+	+
45	GetLastError	+	+	+	+
46	GeneralOperationXML	+	+	+	+
47	GetDeviceLongInfo	+	+	+	
48	SetDeviceLongInfo	+	+	+	
49	ModifyDuressFP	+	+	+	
50	GetMachineIP				
51	GetDepartName		+	+	
52	SetDepartName		+	+	
53	StartEventCapture	+	+	+	+
54	StopEventCapture	+	+	+	+
55	OnReceiveEventXML	+	+	+	+
56	ConnectP2p	+	+	+	+
Methods \ Machines		S100 (SB2900)	S300 (SB2960)	SB3600	SB2910 (EC500)

Appendix 2. Miscellaneous Items

- OCX registration in WIN7



- How to view logged photo in SB3600