

PullSDK Interfaces User Guide

Version: 2.0

Support PullSDK 2.2.0.169 and above version

Date: Jan. 2012

About This Manual

This User Guide introduces the interface and function operations of the PullSDK.

Contents

1. Overview of the PullSDK Interfaces.....	- 1 -
2. Description of the PullSDK Interface Technology.....	- 1 -
3. Installation of the PullSDK Interface.....	- 1 -
4. Detailed Description of the PullSDK Interface Functions.....	- 2 -
4.1 Connect.....	- 2 -
4.2 Disconnect.....	- 3 -
4.3 SetDeviceParam.....	- 3 -
4.4 GetDeviceParam.....	- 4 -
4.5 ControlDevice.....	- 4 -
4.6 SetDeviceData.....	- 6 -
4.7 GetDeviceData.....	- 6 -
4.8 GetDeviceDataCount.....	- 8 -
4.9 DeleteDeviceData.....	- 9 -
4.10 GetRTLog.....	- 10 -
4.11 SearchDevice.....	- 11 -
4.12 ModifyIPAddress.....	- 12 -
4.13 PullLastError.....	- 13 -
4.14 SetDeviceFileData.....	- 14 -
4.15 GetDeviceFileData.....	- 15 -
4.16 ProcessBackupData.....	- 16 -
5. Appendix.....	- 17 -
5.1 Attached Table 1: Detailed Description of Interface Files.....	- 17 -
5.2 Attached Table 2: Description of Controller Parameters.....	- 17 -
5.3 Attached Table 3: Description of ControlDevice Parameters.....	- 24 -
5.4 Attached Table 4: Description of Structure of Function Tables.....	- 25 -
5.5 Attached Table 5: Description of Error Codes in the Returned Values.....	- 28 -
5.6 Attached Table 6: Description of Event Types and Code.....	- 30 -
5.7 Formats of Returned Data in Buffer of the GetRTLog Function.....	- 34 -

1. Overview of the PullSDK Interfaces

The PullSDK interfaces are a group of functions, which are used to access the data of the C3 and C4 access control panels. PullSDK enables the developers of final application programs to access the access control panel more visually, conveniently, and concisely. The PullSDK interface provides the following functions:

Read and set the controller parameters

Read, set, and delete the related information (for example, time segment, user information, and holiday information) of the controller

Search for and modify the device information

2. Description of the PullSDK Interface Technology

In the eyes of the developers of final application programs, the PullSDK interfaces are a group of extract interfaces that are used to set and get the data in the access control panel. It seems that the developers are using the most universal SQL sentences while accessing the user data. In the eyes of the developers of application programs, the PullSDK interfaces seem to be a database server.

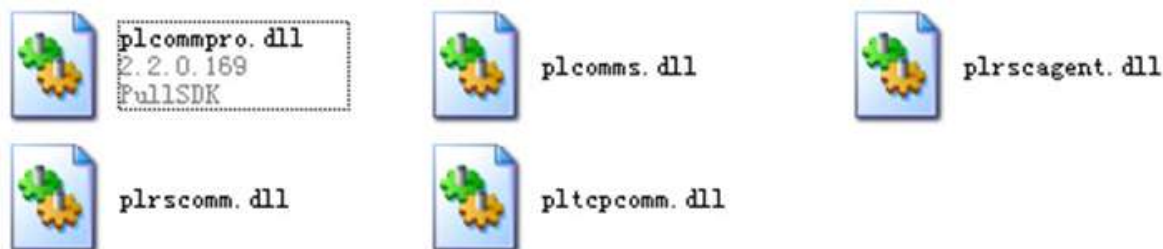
The PullSDK interfaces support the TCP/IP and RS485 communication protocol.

The PullSDK interfaces are developed by using the C language. Data communication is highly optimized, thus turning the PullSDK interfaces into the concise and efficient access interfaces.

Initially, the PullSDK interfaces are designed by referring to the SQL, but the most commonly used service model is the first consideration. Generally, the PullSDK interfaces are a group of elaborately abstracted interfaces, which attain a good balance between design, implementation, and use.

3. Installation of the PullSDK Interface

The PullSDK interface functions are contained in the plcommpro.dll file, which relies upon several other files. You need to copy the following five DLL files together to the system directory under Windows (windows/system32 under Windows XP).



(Note: Attached table 1 describes the functions of every file).

4. Detailed Description of the PullSDK Interface Functions

4.1 Connect

[Function]

`int Connect(const char *Parameters)`

[Objective]

The function is used to connect a device. After the connection is successful, the connection handle is returned.

[Parameter description]

Parameters:

[in]: Specify the connection options through the parameter, for example:

"protocol=RS485,port=COM2,baudrate=38400bps,deviceid=1,timeout=50000,passwd=";

"protocol=TCP,ipaddress=192.168.12.154,port=4370,timeout=4000,passwd=";

To connect a device, the system needs to transfer the device-related connection parameters.

protocol indicates the protocol used for communication. At present, RS485 and TCP can be used.

port: Communication port of the device. For example, if the RS485 protocol is used, you can set **port** to **COM1**: If the TCP is used, the default **port** is **4370** unless otherwise noted.

deviceid: Device ID used by the serial port.

baudrate: Baud rate used for the communication of the communication of the serial port.

ipaddress: IP address of the related device for TCP/IP communication.

timeout: Timeout time of the connection (unit: ms)If the network connection is in poor condition, you should set the parameter to a larger value. Usually, **timeout=5000 (5 seconds)** can meet the basic network needs. When the query result contains the error code of **-2**, you should set **timeout** to a larger value, for example, **timeout=20000 (20 seconds)**.

passwd: Connection password of the communication. If the parameter value is null, it indicates that no password is used.

(Note: The connection parameters are case-sensitive)

[Returned value]

If the device is connected successfully, the connection handle is returned. Otherwise, the error code of **0** is returned.

[Example]

Python:

```
params = "protocol=TCP,ipaddress=192.168.12.154,port=4370,timeout=4000,passwd="
```

```
self.commpo = windll.LoadLibrary("plcommpo.dll")
```

```
constr = create_string_buffer(params)
```

```
self.hcommpo = self.commpo.Connect(constr)
```

c#:

```
params = "protocol=TCP,ipaddress=192.168.12.154,port=4370,timeout=2000,passwd=" ;
```

```
IntPtr h = Connect(params);
```

4.2 Disconnect

[Function]

Void Disconnect(**HANDLE** handle)

[Objective]

The function is used to disconnect the device.

[Parameter description]

handle

[in]: The handle that is returned when the connection is successful.

[Returned value]

None

[Example]

Python:

```
self.commpo.Disconnect(self.hcommpo)
```

```
self.hcommpo = 0
```

c#:

```
Disconnect(h);
```

```
h = IntPtr.Zero;
```

4.3 SetDeviceParam

[Function]

int SetDeviceParam(**HANDLE** handle, **const char** *ItemValues)

[Objective]

The function is used to set the controller parameters, for example, the device ID, door sensor type, driving time of the lock, and read interval.

[Parameter description]

handle

[in]: The handle that is returned when the connection is successful.

Item Values

[in]: The device parameter value to be set; the multiple parameter values can be separated by commas; you can set at most 20 parameters at a time (Attached table 2 lists the parameter value attributes).

[Returned value]

When the returned value is **0** , it indicates that the operation is successful. When the returned value is a negative value, it indicates an error. Attached table 5 lists the information about the error codes.

[Example]

Python:

```
items = ("DeviceID=1,Door1SensorType=2,Door1Drivertime=6,Door1Intertime=3")
```

```
p_items = create_string_buffer(items)
```

```
ret = self.commpo.SetDeviceParam(self.hcommpo, p_items)
```

c#:

```
int ret = 0;
```

```
items = ("DeviceID=1,Door1SensorType=2,Door1Drivertime=6,Door1Intertime=3")
```

```
ret = SetDeviceParam(h, items);
```

4.4 GetDeviceParam

[Function]

`int` GetDeviceParam(`HANDLE` handle, `char` *Buffer, `int` BufferSize, `const char` *Items)

[Objective]

The function is used to read the controller parameters, for example, the device ID, door sensor type, driving time of the lock, and read interval.

[Parameter description]

handle

[in]: The handle that is returned when the connection is successful.

Buffer

[in]: The buffer used to receive the returned data; the returned data is expressed in a text format; if the returned data is multiple params, the multiple params are separated by **commas**.

Buffer Size

[in] The size of the buffer used to receive the returned data.

Items

[in]: The parameter names of the device to be read; the multiple parameter names are separated by commas; you can read at most 30 parameters at a time (Attached table 1 lists the parameter value attributes).

[Returned value]

When the returned value is **0**, it indicates that the operation is successful. When the returned value is a negative value, it indicates that the operation fails. Attached table 5 lists the information about the error codes.

[Example]

Python:

```
buffer = create_string_buffer(2048)
items = ("DeviceID,Door1SensorType,Door1Drivertime,Door1Intertime")
p_items = create_string_buffer(items)
ret=self.commpo.GetDeviceParam(self.hcommpo, buffer, 256, p_items)
```

c#:

```
int ret = 0;
int BUFFERSIZE = 10 * 1024 * 1024;
byte[] buffer = new byte[BUFFERSIZE];
items = ("DeviceID,Door1SensorType,Door1Drivertime,Door1Intertime");
ret = GetDeviceParam(h, ref buffer [0], BUFFERSIZE, items);
```

4.5 ControlDevice

[Function]

`int` ControlDevice(`HANDLE` handle, `LONG` OperationID, `LONG` Param1, `LONG` Param2, `LONG` Param3, `LONG` Param4, `const char` *Options)

[Objective]

The function is used to control the actions of the controller.

[Parameter description]

handle

[in]: The handle that is returned when the connection is successful.

OperationID

[in] Operation contents: 1 for output, 2 for cancel alarm, 3 for restart device, and 4 for enable/disable normal open state.

Param1

[in] When the OperationID is output operation: If Param2 is the door output the parameter indicates the door number. If Param2 is auxiliary output, the parameter indicates the number of the auxiliary output interface (for details, see Attached table 3). If Param2 is cancel alarm, the parameter value is 0 by default.

Param2

[in]: When the OperationID is output operation, this parameter indicates the address type of the output point (1 for door output, 2 for auxiliary output), for details, see Attached table 3. When the OperationID is cancel alarm, the parameter value is 0 by default. When the OperationID value is 4, that is enable/disable normal open state, the parameter indicates is enable/disable normal open state (0 for disable, 1 for enable).

Param3

[in]: When the OperationID is output operation, the parameter indicates the door-opening time (0 indicates the closed state, 255 indicates the normal open state, the value range is 1 to 60 seconds). The default value is 0.

Param4

[in] Reserved; the default value is 0.

Option

[in]: The default value is null; it is used for extension.

[Returned value]

When the returned value is **0** or a positive value, it indicates that the operation is successful. When the returned value is a negative value, it indicates that the operation fails. Attached table 5 lists the information about the error codes.

[Example]

Python:

```
operation_id = 1
door_id = 1
index = 2
state = 6
ret = self.commpro.ControlDevice(self.hcommpro, operation_id, door_id, index, state, 0, '')
```

c#:

```
int ret = 0;
int operid = 1;
int doorid = 0;
int outputadr = 0;
int doorstate = 8;
ret = ControlDevice(h, operid, doorid, outputadr, doorstate, 0, "");
```

4.6 SetDeviceData

[Function]

`int SetDeviceData(HANDLE handle, const char *TableName, const char *Data, const char *Options)`

[Objective]

The function is used to set the device data (for example, the time segment, user information, and holiday information). The device data can be one or multiple records.

[Parameter description]

handle

[in]: The handle that is returned when the connection is successful.

TableName

[in]: Data table name. Attached table 4 lists the available data tables.

Data

[in]: Data record; the data is expressed in a text format; the multiple records are separated by `\r\n`, and the “Field=Value” pairs are separated by `\t`.

Options

[in]: The default value is null; it is used for extension.

[Returned value]

When the returned value is `0`, it indicates that the operation is successful. When the returned value is a negative value, it indicates that the operation fails. Attached table 5 lists the information about the error codes.

[Example]

Python:

```
table = "user"           # User information table
data = "Pin=19999\tCardNo=13375401\tPassword=1\r\nPin=2\tCardNo=14128058\tPassword=1"
p_table = create_string_buffer(table)
str_buf = create_string_buffer(data)
ret = self.commpro.SetDeviceData(self.hcommpro, p_table, str_buf, '')      # Upload the str_buf data
to the user information table
```

c#:

```
int ret = 0;
string devtablename = "user";
string data = "Pin=19999\tCardNo=13375401\tPassword=1\r\nPin=2\tCardNo=14128058\tPassword=1";
string options = "";
ret = SetDeviceData(h, devtablename, data, options);
```

4.7 GetDeviceData

[Function]

`int GetDeviceData(HANDLE handle, char *Buffer, int BufferSize, const char *TableName, const char *FieldNames, const char *Filter, const char *Options)`

[Objective]

The function is used to read the device data (for example, the punch records, time segment, user information, and holiday information). The data can be one or multiple records.

[Parameter description]

handle

[in]: The handle that is returned when the connection is successful.

Buffer

[in]: The buffer used to receive the returned data; the returned data is expressed in a text format; if the returned data is multiple records, the multiple records are separated by `\r\n`.

BufferSize

[in] The size of the buffer used to receive the returned data.

TableName

[in]: Data table name. Attached table 4 lists the available data tables.

FieldNames

[in]: Field name list; the multiple fields are separated by semicolons; * indicates all fields, and the first line in the returned data field is the field names.

Filter

[in]: The conditions of reading the data; the character string in the format of “field name, operator, value” can support multiple conditions, which are separated by commas; for example:

<Field name>=<Value>(no space is allowed at two sides of =)

Options

[in]: Only used to download data of the access control records event effectively at present; when the parameter value is **New Record**, new records are downloaded. When the value is null, all records are downloaded. When download the other table data, this field can set to an empty string.

[Returned value]

When the returned value is **0** or a positive value, it indicates that the operation is successful (the returned value indicates the number of records). When the returned value is a negative value, it indicates that the operation fails. Attached table 5 lists the information about the error codes.

[Example]

Python:

```
table = "user"          # Download the user data from the user table
fieldname = "*"         # Download all field information in the table
pfilter = ""           # Have no filtering conditions and thus download all information
options = ""

query_buf = create_string_buffer(4*1024*1024)
query_table = create_string_buffer(table)
query_fieldname = create_string_buffer(fieldname)
query_filter = create_string_buffer(filter)
query_options = create_string_buffer(options)
ret = self.commpo.GetDeviceData(self.hcommpo, query_buf, 4*1024*1024, query_table,
query_fieldname, query_filter, query_options)
```

c#:

```
int ret = 0;
int BUFFERSIZE = 10 * 1024 * 1024;
byte[] buffer = new byte[BUFFERSIZE];
string devtablename = "user";
string str = "*";
string devdatfilter = "";
```

```
string options = "";
ret = GetDeviceData(h, ref buffer[0], BUFFERSIZE, devtablename, str, devdatfilter, options);
```

4.8 GetDeviceDataCount

[Function]

```
int GetDeviceDataCount(void *Handle, const char *TableName, const char *Filter, const char *Options)
```

[Objective]

The function is used to read the total number of records on the device and return the number of records for the specified data.

[Parameter description]

Handle

[in]: The handle that is returned when the connection is successful.

TableName

[in]: Data table name. Attached table 4 lists the available data tables.

Filter

[in]: The default value is null; it is used for extension.

Options

[in]: The default value is null; it is used for extension.

[Returned value]

When the returned value is **0** or a positive value, it indicates that the operation is successful (the returned value indicates the number of records). When the returned value is a negative value, it indicates that the operation fails. Attached table 5 lists the information about the error codes.

[Example]

Python:

```
table = 'user'
filter = ""
p_table = create_string_buffer(table)
p_filter = create_string_buffer(filter)
ret = self.commpo.GetDeviceDataCount(self.hcommpo, p_table, p_filter, "")
```

c#:

```
int ret = 0;
string devtablename = "user";
string devdatfilter = "";
string options = "";
ret = GetDeviceDataCount(h, devtablename, devdatfilter, options);
```

4.9 DeleteDeviceData

[Function]

int DeleteDeviceData(**HANDLE** handle, **const char** *TableName, **const char** *Data, **const char** *Options)

[Objective]

The function is used to delete the data (for example, user information and time segment) on the device.

[Parameter description]

handle

[in]: The handle that is returned when the connection is successful.

TableName

[in]: Data table name. Attached table 4 lists the available data tables.

Data

[in]: Data record; the data is expressed in a text format; the multiple records are separated by `\r\n`, and the “Field=Value” pairs are separated by `\t`.

Options

[in]: The default value is null; it is used for extension.

[Returned value]

When the returned value is **0** or a positive value, it indicates that the operation is successful. When the returned value is a negative value, it indicates that the operation fails. Attached table 5 lists the information about the error codes.

[Example]

Python:

```
table = "user"
data = "Pin=2"          # Conditions of deleting the data
p_table = create_string_buffer(table)
p_data = create_string_buffer(data)
ret = self.commpro.DeleteDeviceData(self.hcommpro, p_table, p_data, "")
```

c#:

```
int ret = 0;
string devtablename = "user";
string data = "Pin=2";
string options = "";
ret = DeleteDeviceData(h, devtablename, data, options);
```

4.10 GetRTLog

[Function]

int GetRTLog([HANDLE](#) handle, [char](#) *Buffer, [int](#) BufferSize)

[Objective]

The function is used to acquire the realtime event records generated by the equipment and the door status or alarm status of the equipment.

[Parameter description]

handle

[in]: The handle that is returned when the connection is successful.

Buffer

[in] The buffer used to receive the returned data, the returned data is expressed in a text format.

This buffer stores two types of data: realtime event records and door/alarm status. The data returned by this function can be only one of these types. A realtime event query can return multiple records simultaneously (which depends on the number of event records in the realtime monitoring buffer on the equipment at that time). For details of data formats in the buffer, see Attachment 7.

BufferSize

[in]: The size of the buffer used to receive the returned data.

[Returned value]

When the returned value is **0** or a positive value, it indicates the number of records for the received data. When the returned value is a negative value, it indicates that the operation fails. Attached table 5 lists the information about the error codes.

[Example]

Python:

```
rt_log = create_string_buffer(256)
ret = self.commpro.GetRTLog(self.hcommpro, rt_log, 256)
```

c#:

```
int ret = 0;
int buffersize = 256;
byte[] buffer = new byte[256];
ret = GetRTLog(h, ref buffer[0], buffersize);
```

4.11 SearchDevice

[Function]

`int SearchDevice(char *CommType, char *Address, char *Buffer)`

[Objective]

The function is used to search for the access control panel in the LAN.

[Parameter description]

CommType

[in]: If the communication type is set to UDP (or Ethernet), all devices of the specified communication type will be searched.

Address

[in]: Broadcast address; the system searches for the devices in the LAN within the specified IP address range; the default value is **255.255.255.255**, known as network broadcasting.

Buffer

[in]: The buffer is used to save the detected devices. Users should determine the requested memory according to the number of devices in the corresponding network. For example, if the network has not more than 50 devices, it is recommended that users should request the memory of 32K; if the network has not more than 100 devices, it is recommended that users should request the memory of 64K.

[Returned value]

When the returned value is **0** or a positive value, it indicates the number of found access control panels. When the returned value is a negative value, it indicates that the operation fails. Attached table 5 lists the information about the error codes.

[Note]

This approach is intended to search for access controllers on a LAN in UDP broadcast mode. UDP packets cannot traverse routers, so an access controller must not be separated from a server by routers. If by this means you find a device that resides on a different network segment as a server but fail to ping the IP address of an access controller, you may set the controller and server addresses to be on the same subnet (not necessarily on the same network segment). For details on network setting, consult related administrator to obtain correct IP addresses, subnet masks and gateways.

[Example]

Python:

```
dev_buf = create_string_buffer("", 64*1024)
ret=self.commpro.SearchDevice("UDP", "255.255.255.255", dev_buf)
```

c#:

```
int ret = 0;
string udp = "UDP";
string adr = "255.255.255.255";
byte[] buffer = new byte[64 * 1024];
ret = SearchDevice(udp,adr, ref buffer[0]);
```

4.12 ModifyIPAddress

[Function]

int ModifyIPAddress(**char** *CommType,**char** *Address, **char** *Buffer)

[Objective]

The function is used to modify the IP addresses of controllers through the UDP broadcast method. For security purposes, only the IP addresses, subnets, and gateways of the access controllers with no passwords specified can be modified.

[Parameter description]

CommType

[in]: Communication modes employed in search of access controllers: UDP (or Ethernet) in this example.

Address

[in]: Broadcast address; the default value is 255.255.255.255.

Buffer

[in]: The buffer is used to save the MAC addresses and new IP addresses of the target device.

Configure subnet masks and gateways according to current network, except for IP addresses.

[Returned value]

When the returned value is **0** or a positive value, it indicates the number of records for the received data. When the returned value is a negative value, it indicates that the operation fails. Attached table 5 lists the information about the error codes.

[Example]

Python:

```
mac = '00:17:61:01:88:27'           # MAC address of the target device
new_ip = '192.168.12.156'           # New IP address of the device
comm_pwd = ""
str = "MAC=%s,IPAddress=%s " % (mac,new_ip)
p_buf = create_string_buffer(str)
modify_ip = self.commpro.ModifyIPAddress("UDP", "255.255.255.255", p_buf)
```

c#:

```
int ret = 0;
string udp = "UDP";
string address = "255.255.255.255";
string buffer = "MAC=00:17:61:01:88:27" + "," + "IPAddress=192.168.12.156";
ret = ModifyIPAddress(udp,address,buffer);
```

4.13 PullLastError

[Function]

int PullLastError()

[Objective]

The function is used to Obtain the returned error code. If an error code return fails by using other error codes, this function can be called to obtain the error code. For example, if 0 is returned when an equipment connection fails by calling **Connect()**, you can run this function to obtain current error code.

[Parameter description]

None

[Returned value]

Error ID.

[Example]

Python:

See the new contents below.

```
params= u"protocol=TCP,ipaddress=192.168.1.201,port=4370,timeout=3000,passwd=123abc"
```

```
constr = create_string_buffer(params)
```

```
self.hcommpro = self.commpo.Connect(constr)
```

```
if self.hcommpro > 0:
```

```
self.connected = True
```

```
else:
```

```
    error = self.commpo.PullLastError()
```

c#:

```
int ret = 0;           // Error code
```

```
string str = "
```

```
    protocol=TCP,ipaddress=192.168.1.201,port=4370,timeout=3000,passwd=123abc ";
```

```
h = Connect(str);
```

```
if (h != IntPtr.Zero)
```

```
{
```

```
    MessageBox.Show("Connect device succeed!");
```

```
}
```

```
else
```

```
{
```

```
    ret = PullLastError();
```

```
    MessageBox.Show("Connect device Failed! The error code is: " + ret);
```

```
}
```

4.14 SetDeviceFileData

[Function]

`int SetDeviceFileData(void *Handle, const char *FileName, char *Buffer, int BufferSize, const char *Options)`

[Objective]

The function is used to transfer a file from the PC to the device. It mainly used to transfer the update file. The update file name is emfw.cfg.

[Parameter description]

Handle

[in]: The handle that is returned when the connection is successful.

FileName

[in]: The name of the file transferred to the device, for example, a **emfw.cfg** file.

Buffer

[in]: The data buffer used to transfer a file.

BufferSize

[in] Length of the transferred data.

Options

[in]: The default value is null; it is used for extension.

[Returned value]

When the returned value is **0** or a positive value, it indicates that the operation is successful. When the returned value is a negative value, it indicates that the operation fails. Attached table 5 lists the information about the error codes.

[Example]

Python:

```
file_name = "emfw.cfg"
buff_len = len(file_name)
pfile_name = create_string_buffer(file_name)
pbuffer = create_string_buffer(buff_len)
ret = self.commpro.SetDeviceFileData(self.hcommpro, pfile_name, pbuffer, buff_len, "")
```

c#:

```
int ret = 0;
string filename = "emfw.cfg ";
FileStream fsFile = File.OpenRead(this.openFileDialog1.FileName);
string buffersize = (int)fsFile.Length;
byte[] buffer = new byte[buffersize];
string options = "";
ret = SetDeviceFileData(h, filename, ref buffer[0], buffersize, options);
```


4.15 GetDeviceFileData

[Function]

`int GetDeviceFileData(void *Handle, char *Buffer, int *BufferSize, const char *FileName, const char *Options)`

[Objective]

The function is used to obtain a file from the device to the PC. It can obtain user file, record file and etc.

[Parameter description]

Handle

[in]: The handle that is returned when the connection is successful.

FileName

[in] The name of the file obtained from the device, for example, the user file's name is user.dat, record file's name is transaction.dat.

Buffer

[in]: Buffer used to receive the data.

BufferSize

[in]: Length of the received data.

Options

[in]: The default value is null; it is used for extension.

[Returned value]

When the returned value is **0** or a positive value, it indicates that the operation is successful. When the returned value is a negative value, it indicates that the operation fails. Attached table 5 lists the information about the error codes.

[Example]

Python:

```
file_name = "user.dat"
pfile_name = create_string_buffer(file_name)
pbuffer = create_string_buffer(4*1024*1024)
ret = self.commpro.GetDeviceFileData(self.hcommpro, pbuffer, buff_len, pfile_name, "")
```

c#:

```
int ret = 0;
int buffersize = 4 * 1024 * 1024;
byte[] buffer = new byte[buffersize];
string filename = "user.dat";
string options = "";
ret = GetDeviceFileData(h, ref buffer[0], ref buffersize, filename, options);
```

4.16 ProcessBackupData

[Function]

`int ProcessBackupData(const unsigned char *revBuf, int fileLen, char *outBuf, int outSize)`

[Objective]

The files used for processing equipment backup files, for example, backup files in an SD card.

[Parameter description]

revBuf

[in] The uploaded files;

fileLen

[in] The file length;

outBuf

[in] To receive the returning data;

outsize

[in] The max length of the receiving data.

[Returned value]

The returning value is 0 or positive number for success operation. Otherwise, the operation is failed. For the error codes, please refer to the Appendix 5.

[Example]

Python :

```
filename = "sddata.dat"
```

```
buff_len = len(filename)
```

```
buf = create_string_buffer(filename)
```

```
buffer = create_string_buffer(16*1024*1024)
```

```
ret = self.commpo.ProcessBackupData(buf, buff_len, ref buffer[0], 16 * 1024 * 1024)
```

c# :

```
byte[] buffer = new byte[16 * 1024 * 1024];
```

```
byte[] buf = new byte[16 * 1024 * 1024];
```

```
int BufferSize = 0;
```

```
int ret = -1;
```

```
string filename = "user.dat";
```

```
StreamReader proFile = new StreamReader(filename);
```

```
BufferSize = proFile.BaseStream.Read(buf, 0, 16 * 1024 * 1024);
```

```
ret = ProcessBackupData(buf, BufferSize, ref buffer[0], 16 * 1024 * 1024);
```

5. Appendix

5.1 Attached Table 1: Detailed Description of Interface Files

File Name	Description
plcommpro.dll	Dynamic connection database interface of the PullSDK function
plcomms.dll	Database on which the PullSDK interfaces rely
plrscomm.dll	Database on which the PullSDK interfaces rely
pltcpcomm.dll	Database on which the PullSDK interfaces rely
rscagent.dll	Database on which the PullSDK interfaces rely

5.2 Attached Table 2: Description of Controller Parameters

Attribute Name	Parameter	Read/Write Type	Remarks
SerialNumber of device	~SerialNumber	Read only	
Number of doors	LockCount	Read only	The number of locks on the control board.
Number of readers	ReaderCount	Read only	Only indicates the number of Wiegand standard readers.
Customized input quantity	AuxInCount	Read only	
Customized output quantity	AuxOutCount	Read only	
Communication Password	ComPwd	Read/write	Default: null character string. Maximum: 15-bit characters (including digits and letters).
IP Address	IPAddress	Read/write	Default: 192.168.1.201
Gateway	GATEIPAddress	Read/write	Default value is IPAddress
BaudRate	RS232BaudRate	Read/write	Default: 38400
Subnet mask	NetMask	Read/write	Default: 255.255.255.0
Anti-passback rule (Door 1 and Door 2 each other is anti-passback, when Door 2 will be opened before Door 1 has opened , and Door 1 can't open two consecutive door)	AntiPassback	Read/write	One-door and two-way controller 1:Enable the anti-passback function between the readers of Door 1 Two-door and single-way controller 1: Enable the anti-passback function between Door 1 and Door 2 Two-door and two-way controller 1: Enable the anti-passback function between the readers of Door 1

Attribute Name	Parameter	Read/Write Type	Remarks
SerialNumber of device	~SerialNumber	Read only	<p>2: Enable the anti-passback function between the readers of Door 2</p> <p>3: Enable the anti-passback function between the readers of Door 1 and between the readers of Door 2 respectively</p> <p>4: Enable the anti-passback function between Door 1 and Door 2</p> <p>Four-door and single-way controller</p> <p>1: Enable the anti-passback function between Door 1 and Door 2</p> <p>2: Enable the anti-passback function between Door 3 and Door 4</p> <p>3: Enable the anti-passback function between Door 1 and Door 2, and between Door 3 and Door 4</p> <p>4: Enable the anti-passback function between Door 1,2 and Door 3,4</p> <p>5: Enable the anti-passback function between Door 1 and Door 2,3</p> <p>6: Enable the anti-passback function between Door 1 and Door 2,3,4</p> <p>16: denotes that only supports anti-passback function between the readers of Door 1</p> <p>32: denotes that only supports anti-passback function between the readers of Door 2</p> <p>64: denotes that only supports anti-passback function between the readers of Door 3</p> <p>128: denotes that only supports anti-passback function between the readers of Door 4</p> <p>Other options:</p> <p>48: denotes that Door1 and 2 support concurrent anti-passback among their respective readers.</p> <p>80: denotes that Door1 and 3 support concurrent anti-passback among their respective readers.</p>

Attribute Name	Parameter	Read/Write Type	Remarks
SerialNumber of device	~SerialNumber	Read only	<p>144: denotes that Door1 and 4 support concurrent anti-passback among their respective readers.</p> <p>96: denotes that Door2 and 3 support concurrent anti-passback among their respective readers.</p> <p>160: denotes that Door2 and 4 support concurrent anti-passback among their respective readers.</p> <p>196: denotes that Door3 and 4 support concurrent anti-passback among their respective readers.</p> <p>112: denotes that Door1, 2, and 3 support concurrent anti-passback among their respective readers.</p> <p>176: denotes that Door1, 2, and 4 support concurrent anti-passback among their respective readers.</p> <p>208: denotes that Door1, 3, and 4 support concurrent anti-passback among their respective readers.</p> <p>224: denotes that Door2, 3, and 4 support concurrent anti-passback among their respective readers.</p> <p>240: denotes that Door1, 2, 3 and 4 support concurrent anti-passback among their respective readers.</p> <p>(Choose and configure the preceding options as required)</p>
<p>Interlock</p> <p>(Door 1 and Door 2 each other is interlock. When the Door 1 in the opening , the Door 2 can only be turned off. Instead the Door 2 is opened, the Door 1 can not be opened.)</p>	InterLock	Read/write	<p>Two-door controller</p> <p>1: Interlock Door 1 and Door 2 mutually</p> <p>Four-door control</p> <p>1: Interlock Door 1 and Door 2 mutually</p> <p>2: Interlock Door 3 and Door 4 mutually</p> <p>3: Interlock Door 1, Door 2 and Door 3 mutually</p> <p>4: Interlock Door 1 and Door 2</p>

Attribute Name	Parameter	Read/Write Type	Remarks
SerialNumber of device	~SerialNumber	Read only	
			mutually, and interlock Door 3 and Door 4 mutually 5: Interlock Door 1, Door 2, Door 3 and Door 4 mutually
Duress Password	Door1ForcePassWord Door2ForcePassWord Door3ForcePassWord Door4ForcePassWord	Read/write	Max: 8 digits
Emergency Password	Door1SupperPassWord Door2SupperPassWord Door3SupperPassWord Door4SupperPassWord	Read/write	Max: 8 digits
Lock at door closing	Door1CloseAndLock Door2CloseAndLock Door3CloseAndLock Door4CloseAndLock	Read/write	1: Enabled 0: Disabled
Door sensor type	Door1SensorType Door2SensorType Door3SensorType Door4SensorType	Read/write	0: Not available 1: Normal open 2: Normal closed
Lock driver time length	Door1Drivertime Door2Drivertime Door3Drivertime Door4Drivertime	Read/write	The value range is 0 to 255. 0: Normal closed 255: Normal open 1 to 254: Door-opening duration
Timeout alarm duration of door magnet	Door1Detectortime Door2Detectortime Door3Detectortime Door4Detectortime	Read/write	The value range is 0 to 255. Unit: second
Verify mode	Door1VerifyType Door2VerifyType Door3VerifyType Door4VerifyType	Read/write	1:Fingerprint 4: Card 6:Card or fingerprint 10:Card and fingerprint 11: Card and password
Multi-card opening (The Door can be opened by more than one person through	Door1MultiCardOpenDoor Door2MultiCardOpenDoor	Read/write	0: Disabled 1: Enabled

Attribute Name	Parameter	Read/Write Type	Remarks
SerialNumber of device	~SerialNumber	Read only	
verified by, In Attached table 4< Multi-card opening table > set group number of multi-card to open the door , Person in the group which is more than one authentication, set most five people.)	Door3MultiCardOpenDoor Door4MultiCardOpenDoor		
Opening the door through the first card	Door1FirstCardOpenDoor Door2FirstCardOpenDoor Door3FirstCardOpenDoor Door4FirstCardOpenDoor	Read/write	0: Disabled 1: First-card normal open
Active time segment of the door (time segment in which a valid punch)	Door1ValidTZ Door2ValidTZ Door3ValidTZ Door4ValidTZ	Read/write	Default: 0 (the door is not activated)
Normal-open time segment of the door	Door1KeepOpenTimeZone Door2KeepOpenTimeZone Door3KeepOpenTimeZone Door4KeepOpenTimeZone	Read/write	Default: 0 (the parameter is not set)
Punch interval	Door1Intertime Door2Intertime Door3Intertime Door4Intertime	Read/write	0 means no interval (unit: second)
MCU Watchdog	WatchDog	Read/write	0: Disabled 1: Enabled
4 doors turn 2 doors	Door4ToDoor2	Read/write	0: Disabled 1: Enabled
The date of Cancel Normal Open	Door1CancelKeepOpenDay Door2CancelKeepOpenDay Door3CancelKeepOpenDay	Read only	

Attribute Name	Parameter	Read/Write Type	Remarks
SerialNumber of device	~SerialNumber	Read only	
	Door4CancelKeepOpen Day		
The time of backup SD card	BackupTime	Read/write	The value range are 1 to 24
Reboot the device	Reboot	Write only	Reboot=1
Synchronization time	DateTime	Write only	<p>DateTime= ((Year-2000)*12*31 + (Month-1)*31 + (Day-1))*(24*60*60) + Hour* 60 *60 + Minute*60 + Second;</p> <p>For example, the now datetime is 2010-10-26 20:54:55, so DateTime= 347748895;</p> <p>And calculate the reverse “DateTime = 347748895”;</p> <p>Second = DateTime % 60 ;</p> <p>Minute = (DateTime / 60) % 60 ;</p> <p>Hour = (DateTime / 3600) % 24 ;</p> <p>Day = (DateTime / 86400) % 31 + 1 ;</p> <p>Month= (DateTime / 2678400) % 12 + 1 ;</p> <p>Year = (DateTime / 32140800) + 2000 ;</p>
Door4 turn to Door2	Door4ToDoor2	Read/write	0: Disabled 1: Enabled
One-way / two-way Reader	InBIOTowWay	Read/write	0: One-way 1: Two-way
Device fingerprint identification version	~ZKFPVersion	Read only	9: 9.0 version 10: 10.0 version
Display parameters of daylight saving time	~DSTF	Read/write	0: Never Show(default) 1: show
Enablement parameters of daylight saving time	DaylightSavingTimeOn	Read/write	0: Never start(default) 1: start
Enable mode of daylight saving time	DLSTMode	Read/write	0: mode 1 1: mode 2
Mode of daylight saving time — Start time	DaylightSavingTime	Read/write	The value have 4 bytes: “month-date-hour-minute”
Mode of daylight saving time — Over time	StandardTime	Read/write	The value have 4 bytes: “month-date-hour-minute”
Mode 2 of daylight saving time : Month	WeekOfMonth1	Read/write	The value range are 1 to 12

Attribute Name	Parameter	Read/Write Type	Remarks
SerialNumber of device	~SerialNumber	Read only	
The begin week of daylight saving time Mode 2: XX week	WeekOfMonth2	Read/write	The value range are 1 to 6
The begin day of daylight saving time Mode 2 : XX (Sunday to Saturday)	WeekOfMonth3	Read/write	The value range are 1 to 7
The begin of daylight saving time Mode 2 : Hour	WeekOfMonth4	Read/write	The value range are 0 to 23
The begin of daylight saving time Mode 2 : minute	WeekOfMonth5	Read/write	The value range are 0 to 59
The end of daylight saving time Mode 2 : Month	WeekOfMonth6	Read/write	The value range are 1 to 12
The end week of daylight saving time Mode 2 : XX week	WeekOfMonth7	Read/write	The value range are 1 to 6
The end day of daylight saving time Mode 2 : XX(Sunday to Saturday)	WeekOfMonth8	Read/write	The value range are 1 to 7
The end of daylight saving time Mode 2 : Hour	WeekOfMonth9	Read/write	The value range are 0 to 23
The end of daylight saving time Mode 2 : Minute	WeekOfMonth10	Read/write	The value range are 0 to 59

5.3 Attached Table 3: Description of ControlDevice Parameters

OperationID	Description	Param1	Param2	Param3	Param4	Options
1	Output operation	Door number or auxiliary output number	1: Door output 2: auxiliary output (the address type of output operation)	0: disable 255: normal open state 1~60: normal open or the duration of normal open (If Param2=1, the value of Param3 makes sense)	reserved	Expansion parameter is null
2	Cancel alarm	0 (null)	0 (null)	0 (null)	reserved	Expansion parameter is null
3	Restart device	0 (null)	0 (null)	0 (null)	reserved	Expansion parameter is null
4	Enable/disable normal open state	Door number	0: disable 1: enable	0 (null)	reserved	Expansion parameter is null

Note: If OperationID=1, Param2 determine the Param1 value is door number or auxiliary output number. If Param1 is door number, the max value is the door number that the device permitted. If the Param1 is auxiliary output number, the max value is the auxiliary output number that the device permitted.

5.4 Attached Table 4: Description of Structure of Function Tables

Table Name	TableName	Field	Remarks
Card number information table	user	CardNo, Pin, Password, Group, StartTime, EndTime	The StartTime and EndTime should be specified in a correct format. YYYYMMDD; for example: 20100823; Group indicates the personnel group of multi-card verification.
Access privilege list	userauthorize	Pin, AuthorizeTimezoneId, AuthorizeDoorId	<p>AuthorizeDoorId is authorized by the door:</p> <p>1 denotes LOCK1</p> <p>2 denotes LOCK2</p> <p>3 denotes LOCK1 and LOCK2</p> <p>4 denotes LOCK3</p> <p>5 denotes LOCK1 and LOCK3</p> <p>6 denotes LOCK2 and LOCK3</p> <p>7 denotes LOCK1, LOCK2, LOCK3</p> <p>8 denotes LOCK4</p> <p>9 denotes LOCK1 and LOCK4</p> <p>10 denotes LOCK2 and LOCK4</p> <p>11 denotes LOCK1, LOCK2, LOCK4</p> <p>12 denotes LOCK3 and LOCK4</p> <p>13 denotes LOCK1, LOCK3, LOCK4</p> <p>14 denotes LOCK2, LOCK3, LOCK4</p> <p>15 denotes LOCK1, LOCK2, LOCK3 and LOCK4.</p> <p>Assume that four doors are numbered 1, 2, 3, and 4 respectively, then:</p> $1 \ll (1-1) + 1 \ll (2-1) + 1 \ll (3-1) + 1 \ll (4-1) = 15$ <p>or $(1111)_2 = (15)_{10}$</p>
Holiday table	holiday	Holiday, HolidayType, Loop	<p>The HolidayType value can be 1, 2, and 3.</p> <p>Loop value: 1 (loop by year), 2 (not loop by year)</p>

Table Name	TableName	Field	Remarks
Time zone table	timezone	TimeZoneId, SunTime1, SunTime2, SunTime3, MonTime1, MonTime2, MonTime3, TueTime1, TueTime2, TueTime3, WedTime1, WedTime2, WedTime3, ThuTime1, ThuTime2, ThuTime3, FriTime1, FriTime2, FriTime3, SatTime1, SatTime2, SatTime3, Hol1Time1, Hol1Time2, Hol1Time3, Hol2Time1, Hol2Time2, Hol2Time3, Hol3Time1, Hol3Time2, Hol3Time3	<p>The Time format is as follows: $(\text{hour} * 100 + \text{minute}) \ll 16 + (\text{hour} * 100 + \text{minute})$</p> <p>For example: set 8:30–12:30 on Monday as time segment 1, so the value is MonTime1=54396110: 8:30 $\rightarrow 8 * 100 + 30 \rightarrow 33\text{E}$ (Hex) 12:30 $\rightarrow 12 * 100 + 30 \rightarrow 4\text{CE}$ (Hex) 033E04CE \rightarrow 54396110 (Decimal)</p>
Access control record table	transaction	Cardno, Pin, Verified, DoorID, EventType, InOutState, Time_second	<p>The Verified mode can be as follows: 1: Only finger 3: Only password 4: Only card 11: Card and password 200: Others</p> <p>Time_second should be specified in a correct format: YYYY-MM-DD hh:mm:ss</p> <p>(After writing, data formats will conversion, if want to take out to analysis, analytical formula is as follows:</p> $\begin{aligned} \text{second} &= t \% 60; \\ t & /= 60; \\ \text{minute} &= t \% 60; \\ t & /= 60; \\ \text{hour} &= t \% 24; \\ t & /= 24; \\ \text{day} &= t \% 31 + 1; \\ t & /= 31; \\ \text{month} &= t \% 12 + 1; \\ t & /= 12; \\ \text{year} &= t + 2000; \end{aligned}$ <p>The EventType, See Attachment 6</p>

Table Name	TableName	Field	Remarks
First-card door-opening table	firstcard	Pin, DoorID, TimezoneID	
Multi-card opening table	multicard	Index, DoorId, Group1, Group2, Group3, Group4, Group5	Group 1 to Group 5 are the numbers of the multi-card opening groups
<p>Linkage control I/O table</p> <p>(When the trigger condition is detected and immediately start the other events)</p> <p>For example: Open the door 1 is detected (trigger conditions), the immediate alarm, open the video surveillance, close the door 2, door 3, door 4, etc. (other events)</p>	inoutfun	Index, EventType, InAddr, OutType, OutAddr, OutTime, Reserved	<p>For details on EventTypes, see the types of access control record lists.</p> <p>When the EventType value is 220 (the auxiliary input point is off) or 221 (the auxiliary input point is short-circuited), the input point is the auxiliary input. When the EventType value is not 220 or 221, the input point is a door.</p> <p>The input point InAddr is a door:</p> <p>0: Any door 1: Door 1 2: Door 2 3: Door 3 4: Door 4</p> <p>The input point InAddr is the auxiliary input:</p> <p>0: Any auxiliary input 1: Auxiliary input 1 2: Auxiliary input 2 3: Auxiliary input 3 4: Auxiliary input 4</p> <p>When the OutType value is 0, the output point OutAddr indicates a lock:</p> <p>1: Lock 1 2: Lock 2 3: Lock 3 4: Lock 4</p> <p>When the OutType value is 1, the output point OutAddr indicates the auxiliary output:</p> <p>1: Auxiliary output 1 2: Auxiliary output 2 3: Auxiliary output 3 4: Auxiliary output 4 5: Auxiliary output 5 6: Auxiliary output 6</p>

Table Name	TableName	Field	Remarks
templatev10 table	templatev1 0	Size 、 UID 、 PIN 、 Fingerl D 、 Valid、 Template 、 Resv erd 、 EndTag	

Note: The fields in the table are case-sensitive.

5.5 Attached Table 5: Description of Error Codes in the Returned Values

(1) Error Code of PullSDK and Firmware By provided

Error Code	Description
-1	The command is not sent successfully
-2	The command has no response
-3	The buffer is not enough
-4	The decompression fails
-5	The length of the read data is not correct
-6	The length of the decompressed data is not consistent with the expected length
-7	The command is repeated
-8	The connection is not authorized
-9	Data error: The CRC result is failure
-10	Data error: PullSDK cannot resolve the data
-11	Data parameter error
-12	The command is not executed correctly
-13	Command error: This command is not available
-14	The communication password is not correct
-15	Fail to write the file
-16	Fail to read the file
-17	The file does not exist
-99	Unknown error
-100	The table structure does not exist
-101	In the table structure, the Condition field does not exit
-102	The total number of fields is not consistent
-103	The sequence of fields is not consistent
-104	Real-time event data error
-105	Data errors occur during data resolution.
-106	Data overflow: The delivered data is more than 4 MB in length
-107	Fail to get the table structure
-108	Invalid options
-201	LoadLibrary failure
-202	Fail to invoke the interface
-203	Communication initialization fails
-206	Start of a serial interface agent program fails and the cause generally relies in inexistence or occupation of the serial interface.
-301	Requested TCP/IP version error

Error Code	Description
-302	Incorrect version number
-303	Fail to get the protocol type
-304	Invalid SOCKET
-305	SOCKET error
-306	HOST error
-307	Connection attempt failed

(2) WinSocket Error Codes

10035	<p>Resources temporarily unavailable.</p> <p>This error is returned from operations on nonblocking sockets that cannot be completed immediately, for example, <code>recv</code> (Wsapiref_2i9e.asp) when no data is queued to be read from the socket. It is a non-fatal error, and the operation should be retried later. It is normal for <code>WSAEWOULDBLOCK</code> to be reported as the result from calling <code>connect</code> on a nonblocking <code>SOCK_STREAM</code> socket (Wsapiref_8m7m.asp), since some time must elapse for the connection to be established.</p>
10038	<p>An operation was attempted on something that is not a socket. Either the socket handle parameter did not reference a valid socket, or for <code>select</code>, a member of an <code>fd_set</code> was no valid.</p>
10054	<p>Connection reset by peer.</p> <p>An existing connection was forcibly closed by the remote host. This normally results if the peer application on the remote host is suddenly stopped, the host is rebooted, the host or remote network interface is disabled, or the remote host uses a hard close (See <code>setsockopt</code> (Wsapiref_94aa.asp) for more information on the <code>SO_LINGER</code> option on the remote socket). This error may also result if a connection was broken due to keep-alive activity detecting a failure while one or more operations are in progress. Operations that were in progress fail with <code>WSAENETRESET</code>. Subsequent operations fail with <code>WSAECONNRESET</code>.</p>
10060	<p>Connection timed out.</p> <p>A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond.</p>
10061	<p>Connection refused.</p> <p>No connection could be made because the target machine actively refused it. This usually results from trying to connect to a server that is inactive on the foreign host — that is, one with no server application running.</p>
10065	<p>No route to host.</p> <p>A socket operation was attempted to an unreachable host. See <code>WSAENETUNREACH</code>.</p>

5.6 Attached Table 6: Description of Event Types and Code

Code	Event Types	Description
0	Normal Punch Open	In [Card Only] verification mode, the person has open door permission punch the card and triggers this normal event of open the door.
1	Punch during Normal Open Time Zone	At the normally open period (set to normally open period of a single door or the door open period after the first card normally open), or through the remote normal open operation, the person has open door permission punch the effective card at the opened door to trigger this normal events.
2	First Card Normal Open (Punch Card)	In [Card Only] verification mode, the person has first card normally open permission, punch card at the setting first card normally open period but the door is not opened, and trigger the normal event.
3	Multi-Card Open (Punching Card)	In [Card Only] verification mode, multi-card combination can be used to open the door. After the last piece of card verified, the system trigger this normal event.
4	Emergency Password Open	The password (also known as the super password) set for the current door can be used for door open. It will trigger this normal event after the emergency password verified.
5	Open during Normal Open Time Zone	If the current door is set a normally open period, the door will open automatically after the setting start time, and trigger this normal event.
6	Linkage Event Triggered	When the linkage setting the system takes effect, trigger this normal event.
7	Cancel Alarm	When the user cancel the alarm of the corresponding door, and the operation is success, trigger this normal event.
8	Remote Opening	When the user opens a door from remote and the operation is successful, it will trigger this normal event.
9	Remote Closing	When the user close a door from remote and the operation is successful, it will trigger this normal event.
10	Disable Intraday Normal Open Time Zone	When the door is in Normally Open (NO) state, swipe your valid card five times through the reader or call ControlDevice to disable the NO period on that day. In this case, trigger this normal event.
11	Enable Intraday Normal Open Time Zone	When the door's NO period is disabled, swipe your valid card (held by the same user) five times through the reader or call ControlDevice to enable the NO period on that day. In this case, trigger this normal event.
12	Open Auxiliary Output	If the output point address is set to a specific auxiliary

		output point and the action type is set enabled in a linkage setting record, then this normal event will be triggered as long as this linkage setting takes effect.
13	Close Auxiliary Output	Events that are triggered when you disable the auxiliary input through linkage operations or by calling ControlDevice.
14	Press Fingerprint Open	Normal events that are triggered after any person authorized to open the door presses his fingerprint and passes the verification in “Fingerprint only” or “Card/Fingerprint” verification modes.
15	Multi-Card Open (Press Fingerprint)	Multi-card open(Fingerprint required): normal events that are triggered when the last person opens the door with his fingerprint in “Fingerprint” verification mode.
16	Press Fingerprint during Normal Open Time Zone	Normal events that are triggered after any person authorized to open the door presses his valid fingerprint during the NO duration (including the NO durations set for single doors and the first-card NO duration) and through remote operations.
17	Card plus Fingerprint Open	Normal events that are triggered after any person authorized to open the door swipes his card and presses his fingerprint to pass the verification in the “Card + Fingerprint” verification mode.
18	First Card Normal Open (Press Fingerprint)	Normal events that are triggered after any person authorized to open the door becomes the first one to press his fingerprint and pass the verification during the preset first-card NO duration and in either the “Fingerprint only” or the “Card/Fingerprint” verification mode.
19	First Card Normal Open (Card plus Fingerprint)	Normal events that are triggered after any person authorized to open the door becomes the first one to swipe his card and press his fingerprint to pass the verification during the preset first-card NO duration and in the “Card + Fingerprint” verification mode.
20	Too Short Punch Interval	When the interval between two card punching is less than the interval preset for the door, trigger this abnormal event.
21	Door Inactive Time Zone (Punch Card)	In [Card Only] verification mode, the user has the door open permission, punch card but not at the door effective period of time, and trigger this abnormal event.
22	Illegal Time Zone	The user with the permission of opening the current door, punches the card during the invalid time zone, and triggers this abnormal event.
23	Access Denied	The registered card without the access permission of the current door, punch to open the door, triggers this abnormal event.
24	Anti-Passback	When the anti-pass back setting of the system takes effect, triggers this abnormal event.

25	Interlock	When the interlocking rules of the system take effect, trigger this abnormal event.
26	Multi-Card Authentication (Punching Card)	Use multi-card combination to open the door, the card verification before the last one (whether verified or not), trigger this normal event
27	Unregistered Card	Refers to the current card is not registered in the system, trigger this abnormal event.
28	Opening Timeout:	The door sensor detect that it is expired the delay time after opened, if not close the door, trigger this abnormal event
29	Card Expired	The person with the door access permission, punch card to open the door after the effective time of the access control, can not be verified and will trigger this abnormal event.
30	Password Error	Use card plus password, duress password or emergency password to open the door, trigger this event if the password is wrong.
31	Too Short Fingerprint Pressing Interval	When the interval between two consecutive fingerprints is less than the interval preset for the door, trigger this abnormal event.
32	Multi-Card Authentication (Press Fingerprint)	In either the "Fingerprint only" or the "Card/Fingerprint" verification mode, when any person presses his fingerprint to open the door through the multi-card access mode and before the last verification, trigger this event regardless of whether the verification attempt succeeds.
33	Fingerprint Expired	When any person fails to pass the verification with his fingerprint at the end of the access control duration preset by himself, trigger this event.
34	Unregistered Fingerprint	Events that are triggered when any fingerprints are not registered in the system or registered but not synchronized to the device.
35	Door Inactive Time Zone (Press Fingerprint)	Abnormal events that are triggered when any person authorized to open the door presses his fingerprint during the preset valid duration.
36	Door Inactive Time Zone (Exit Button)	Abnormal events that are triggered when any person fails to open the door by pressing the Unlock button during the preset valid duration.
37	Failed to Close during Normal Open Time Zone	Abnormal events that are triggered when any person fails to close the door in NO state by calling ControlDevice .
101	Duress Password Open	Use the duress password of current door verified and triggered alarm event.
102	Opened Accidentally	Except all the normal events (normal events such as user with door open permission to punch card and open the door, password open door, open the door at normally open period, remote door open, the linkage triggered door open), the door sensor detect the door is opened, that is the door is

		unexpectedly opened.
103	Duress Fingerprint Open	Use the duress fingerprint of current door verified and triggered alarm event.
200	Door Opened Correctly	When the door sensor detects that the door has been properly opened, triggering this normal event.
201	Door Closed Correctly	When the door sensor detects that the door has been properly closed, triggering this normal event.
202	Exit button Open	User press the exit button to open the door within the door valid time zone, and trigger this normal event.
203	Multi-Card Open (Card plus Fingerprint)	Normal events that are triggered when any person passes the verification with his card and fingerprint in multi-card access mode.
204	Normal Open Time Zone Over	After the setting normal open time zone, the door will close automatically. The normal open time zone include the normal open time zone in door setting and the selected normal open time zone in first card setting.
205	Remote Normal Opening	Normal events that are triggered when the door is set to the NO state for remote opening operations.
206	Device Start	When the device is being activated, this normal event is triggered.
220	Auxiliary Input Disconnected	When any auxiliary input point breaks down, this normal event is triggered.
221	Auxiliary Input Shorted	When any auxiliary input point has short circuited, this normal event is triggered.
255	Actually that obtain door status and alarm status	See Attachment 7

5.7 Formats of Returned Data in Buffer of the GetRTLog Function

When the data in the buffer is resolved and detected to be:

- Multiple realtime event records: separate those records into single ones with “\r\n”.
- Door and alarm status recorded in single entries: separate those single records with a comma considering that the data of single records is separated with a comma.

When you resolve single records, make adjustments according to bit 4 of the separated data. If bit 4 is 255, this record contains the door status and alarm status only; otherwise, this record contains realtime event records.

The following table compares the data structures of these two records.

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6
Door/Alarm Status	Time	DSS status (0: no DSS; 1: door closed; 2: door open)	Alarm status (1: alarm; 2: door opening timeout)	Temporarily not in use	255	Temporarily not in use	200 (Indicates that the verification mode is “none”); not in use
Realtime Event Records	Time	Pin (Employee No.)	Card No.	Door No., namely lock number	Event type code. See Attachment 6 for details.	Entry/Exit status: (0: entry; 1: exit; 2: none)	The verification mode is the same as the door opening mode of controller parameters described in Attachment 2.

Note:

(1)

The device can temporarily save a maximum of 30 realtime event records. You can call **GetRTLog** to check whether the cache contains event records. If so, the device returns all records (30 entries at most) in the current cache; otherwise, the device returns the door and alarm status events referred above.

(2)

The door status records contain the open/closed status of current door (on the premise that the DSS is connected). Additionally, you can judge the current door status through “Door already open” (Event code: 200) and “Door already closed” (Event code: 201).

(3)

When the record adopts the door/alarm status, the door status contained in all records actually is the door status (four doors at most) of all doors of the device. 4 bytes are respectively represents four door status,

arranged in an ascending order separately represent doors 1 to 4. For example, if this value is 0x01020001, door No.1 is closed, door No.2 is not configured with the DSS, door No.3 is door opened, and door No.4 is door closed. Contained in the alarm status (and Opening Timeout) (The Second place) the same that 4 bytes are respectively represents four door status, behind two place of Each byte respectively represents whether that have alarm or door open is overtime, arranged in an ascending order separately represent alarm or door opening timeout. For example, if this value is 0x01020001, door No.1 is closed, door No.3 means door opening timeout, door No.2 and No.4 means alarm.

(4)

When the record adopts “realtime event” status and type of event is Triggered Linkage Event (the code of type event: 6), the sixth place saved Linkage Event Triggered, and the second is for reuse of Linkage ID, It have software for the device synchronous linkage setting (usually the linkage in the ID value of software end database).