SOAP development Manual

Version: **2.0**

2008 . 3

Contents

1. about SOAP SDK

SOAP (Simple Object Access Protocol) is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. SOAP uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols decentralized, SOAP definite a mechanism which is used to transmits the orders and parameters between an HTTP client and server. SOAP has been designed to be independent of any operating system, programming language or the used object model in the server or client: except the HTTP transmission and other is unrelated. SOAP is simple. Client sent a request to the server, and calls the corresponding object, and then the server returns the results. This message is that XML format, and the news is packaged as one which accord with HTTP protocol. The work of SOAP has been adopted through the underlying structure of the Internet. You do not need any work; it is consistent with any router, firewall or proxy server.

SOAP includes four parts:

1. SOAP envelop, SOAP envelope structure has defined an whole framework that may express any content which the message contain, who process these contents, and these contents may be optional or necessary.

2. SOAP Encoding Rules , define a series of mechanisms that exchange data type of the application ;

3. SOAP RPC (the RPC representation), define an agreements that express remote procedure call and response;

4. SOAP binding, is used in the underlying agreement to exchange information.

Although these four parts belong to one part of SOAP and as a whole the definition, but their function is the intersection of each independent. In particular, envelopes and encoding rules are defined in different XML namespace; this allows a more simple definition.

Two main objective of SOAP is designed or simple and extensible. This means that there are some properties of traditional information systems or distributed object system, which will not be part of the SOAP standard. For example: (Distributed garbage collection), (Box carrying or batching of messages), Objects-by-reference (which requires distributed garbage collection), Activation (which requires objects - by-reference).

SOAP ,a very general protocol, is used in the stack of WEB services to transfer XML messaging, play a very important role in the Internet environment, realize that the system is loosely coupled, cross-platform, and the language has nothing to do with the specific Interface unrelated, but also to provide a reliable Web application to visit. "Software - Software dialogue" approach interchangeable, break the incompatible with the software applications, web sites and equipment between the state to achieve the

"WEB-based seamless integration".

SOAP SDK is a tool to achieve data communication with the standalone fingerprint machine through XML protocol. Can be easily used in the standalone fingerprint machines for performance user information, fingerprints management, card record to download, the fingerprint machine settings.

Main features are:

1, download attendance records from the fingerprint machine

2, download, upload user information from the fingerprint information.

3, download information of the system setting.

2. SOAP SDK installation

2.1 How to install

Install the SoapSDK which Based on the Windows platform

1. Please download SOAPSDK3. 0 from Microsoft's Web site
2. 2. Run soapsdk.exe file, SoapSDK development kits will be installed on a computer.

3. Description of the communication process

The following three steps to complete create SOAP client application

1 - Assign and connect Web server.

2 - Prepare and send information.

3 - Read for returning information from service-.

[Example]

Sent orders to restart equipment to the server, XML format are as follows:

[XML protocols]

Request Xml:

```
<Restart>
        <ArgComKey Xsi:type="xsd:integer"> ComKey </ ArgComKey>
</ Restart>
```

Response Xml:

```
<RestartResponse>
         <Row>
             < Result> Succeed! </ Result>
         </ Row>
            </ RestartResponse>
```

VC + + source code as follows:

```
/ / define the Pointer which is used to connect the server with http protocol
ISoapConnectorPtr Connector = NULL;

Connector.CreateInstance (__uuidof (HttpConnector30));

/ / specified URL of equipment for the parameters IP

Connector-> Property [ "EndPointURL"] = IP;

/ / Connect equipment

Connector-> Connect ();

/ / be ready for sending SOAP Request

Connector-> BeginMessage ();

/ / Build SOAP message , and sent it to the server
```

```
ISoapSerializerPtr Serializer = NULL;

Serializer.CreateInstance (__uuidof (SoapSerializer30));

/ /According to the built-in sequence the MS SOAP Toolkit provide, begin to sent the request

Serializer-> Init (_variant_t ((IUnknown *) Connector-> InputStream));

/ / Begin to deal with SOAP message.

Serializer-> StartEnvelope ("","","");

/ / Start the text

Serializer-> StartBody ("");

/ / process sub-elements of the first layer , the first parameter is the name of element,    get
customer information service order parameters GetUserInfo; second parameter is a URL.

    Serializer-> StartElement ( "GetUserInfo", "http://192.168.1.201/Service/message/ ","","");

/ / process elements of the second layer , the first parameter is the element name, port
parameters ArgComKey; second parameter is a URL.

Serializer-> StartElement ( "ArgComKey", "http:// 192.168.1.201/Service/message/ ","","");

/ / Write element value.

Serializer-> WriteString (Com);

/ / End sub-elemen of the second lay t ArgComKey.

Serializer-> EndElement ();

/ / deal with sub- elements of the second layer , the first parameter is the name of element Arg;
second parameter is a URL.

Serializer-> StartElement ( "Arg ","","","");

/ / sub-elements of third lay, the first parameter is the element name, user No.parameters PIN.
Serializer-> StartElement ( "PIN ","","","");

/ / Write element value.

Serializer-> WriteString ( "All");

/ / End sub-elements of the third layer PIN.

Serializer-> EndElement ();

/ / Third lay of sub-elements, the first parameter is the element name, the parameters of user’s
fingerprint No. FingerID.

Serializer-> StartElement ( "FingerID ","","","");

/ / Write element value.
Serializer-> WriteString ( "All");
```

```
/ / End sub-elements of the third layer FingerID.

Serializer-> EndElement ();

/ / End an sub- element of the second lay Arg.

Serializer-> EndElement ();

/ / End an sub- element of the first layer GetUserInfo.

Serializer-> EndElement ();

/ / Close body.

Serializer-> EndBody ();

/ / End SOAP message.

Serializer-> EndEnvelope ();

/ / Send the message to the server

Connector-> EndMessage ();

/ / If the return value is not null, and receive data

If (Connector-> OutputStream! = NULL)

(
/ / Create SOAPReader object.

ISoapReaderPtr Reader = NULL;

Reader.CreateInstance (__uuidof (SoapReader30));

/ / Connect to outputstream.

Reader-> Load (_variant_t ((IUnknown *) Connector-> OutputStream), "");

/ / Use text attribute to get attribute values of the element.

Reader-> RPCResult-> text;
)
```

4. Operation described in detail

4.1 Data Management

[Note common parameters]

ComKey: communications Password (password corresponding machine links, unless the local communication passwords and machine links passwords is consistent, data on the machines is able to sent to local, local can correctly receive it)

PIN: User ID (Registration).

4.1.1 user information

4.1.1.1 Get the user's information via user enrollment

[Function]

Achieved through user registration, user information

[XML protocols]

Request Xml:

```
        <GetUserInfo>
                <ArgComKey Xsi:type="xsd:integer"> ComKey </ ArgComKey>
                  <Arg>
                        <PIN Xsi:type="xsd:integer"> Job Number </ PIN>
        </ Arg>
        </ GetUserInfo>
```

Response Xml:

```
   <GetUserInfoResponse>
<Row>
        <PIN> XXXXX </ PIN>
        <Name> XXXX </ Name>
        <Password> XXX </ Password>
        X <Group> </ Group>
        X <Privilege> </ Privilege>
        <Card> XXXX </ Card>
```

```
            <PIN2> XXXX </ PIN2>
            <TZ1> XXX </ TZ1>
            <TZ2> XXX </ TZ2>
            <TZ3> XXX </ TZ3>
      </ Row>
      </ GetUserInfoResponse>
```

[Parameters]

ComKey: communications Password

PIN: User ID (Registration).

[Return value]

if successful, the function returns the user's information, else return null.
PIN: 5 station code is same with PIN2 code, nine station code is internal code.
Name: user name.

Password: user passwords.

Group: Group

Privilege: user privileges; 0, 1 ordinary users, administrators.

Card: Number

PIN2: User ID (Registration).

TZ1: TimeZone1

TZ2: TimeZone2

TZ3: TimeZone 3

4.1.1.2 Write into the user's information

[**Function**]

Same as enrollment user, write your personal information.

[XML protocols]

Request Xml:

```
   <GetUserInfo>
         <ArgComKey Xsi:type="xsd:integer"> ComKey </ ArgComKey>
          <Arg>
```

```
                    <PIN> XXXXX </ PIN>
                    <Name> XXXX </ Name>
                     <Password> XXX </ Password>
                     X <Group> </ Group>
                     X <Privilege> </ Privilege>
                     <Card> XXXX </ Card>
                     <PIN2> XXXX </ PIN2>
                     <TZ1> XXX </ TZ1>
                     <TZ2> XXX </ TZ2>
                     <TZ3> XXX </ TZ3>
                </ Arg>
                </ GetUserInfo>
```

Response Xml:

```
          <GetUserInfoResponse>
             <Row>
          <Result> Succeed! </ Result>
             </ Row>
          </ GetUserInfoResponse>
```

[Parameters]

ComKey: communications Password

PIN: User ID (Registration).

[Return value]

If successful, the function returns the user's information, else return null.
PIN: 5 station code is same with PIN2 code, nine station code is internal code.
Name: user name.

Password: user passwords.

Group: Group

Privilege: user privileges; 0, 1 ordinary users, administrators.

Card: Number

PIN2: User ID (Registration).

TZ1: TimeZone1

TZ2: TimeZone2

TZ3: Time Zone 3

[Return value]

If successful return True, or else return to False.

The function is same with enrolment users, but it has not write fingerprint template, if want, can use the function SetUsertTemplate to upload fingerprint template

to delete a user's information

[Function]

Delete a someone of users.

[**XML protocols**]

Request Xml:

```
<DeleteUser>
        <ArgComKey Xsi:type="xsd:integer"> ComKey </ ArgComKey>
            <Arg>
            <PIN Xsi:type="xsd:integer"> Job Number </ PIN>
            </ Arg>
</ DeleteUser>
```

Response Xml:

```
<DeleteUserResponse>
        <Row>
<Result> Succeed! </ Result>
</ Row>
</ DeleteUserResponse>
```

[Parameters]

ComKey: communications Password

PIN: User ID (Registration).

[Return value]

If successful return True, or else return to False.

4.1.1.4 Get all user information

[Function]

Get all customer information.

[XML protocols]

Request Xml:

```
     <GetAllUserInfo>
              <ArgComKey xsi:type="xsd:integer”>ComKey</ArgComKey>
    </GetAllUserInfo>
```

Response Xml:

```
  <GetAllUserInfoResponse>
        <Row>
          <PIN>XXXXX</PIN>
          <Name>XXXX</Name>
          <Password>XXX</Password>
          < Group>X</ Group>
          < Privilege>X</ Privilege>
          <Card>XXXX </Card>
          <PIN2>XXXX </PIN2>
          <TZ1>XXX </TZ1>
          <TZ2>XXX </TZ2>
          <TZ3>XXX </TZ3>
      </Row>
  </GetAllUserInfoResponse>
```

[Parameters]

ComKey: communications Password

[Return value]

If Successful, return the user's information, else return null.

PIN: 5 station codes are same with PIN2 code, nine station codes is internal code.

Name: user name.

Password: user passwords.

Group: Group

Privilege: user privileges; 0, 1 ordinary users, administrators.

Card: Number

PIN2: User ID (Registration).

TZ1: TimeZone1

TZ2: TimeZone 2

TZ3: TimeZone 3

4.1.1.5 Delete user passwords

[Function]

Remove user passwords.

[XML protocols]

Request Xml:

```
<ClearUserPassword>
        <ArgComKey xsi:type="xsd:integer”>ComKey</ArgComKey>
        <Arg>
        <PIN xsi:type="xsd:integer”>Number</PIN>
        </Arg>
</ClearUserPassword>
```

Response Xml:

```
<ClearUserPasswordResponse>
        <Row>
            <Result>Succeed! </Result>
     </Row>
</ClearUserPassword>
```

[Parameters]

ComKey: communications Password

PIN: User ID (Registration).

[Return value]

If successful, return to True, or else return to False.

4.1.1.6 Delete all user information

[Function]

Delete all user information.

[XML protocols]

Request Xml:

```
<ClearData>
        <ArgComKey xsi:type="xsd:integer”>ComKey</ArgComKey>
        <Arg>
        <Value xsi:type="xsd:integer”>3</Value>
        </Arg>
</ClearData>
```

Response Xml:

```
<ClearDataResponse>
       <Row>
          <Result>Succeed! </Result>
    </Row>
</ClearData>
```

[Parameters]

ComKey: communications Password

Value: Value operation

[Return value]

if successful return to True, or else return to False.

4.1.2 Fingerprints management

4.1.2.1 Get information of user’s Fingerprint template

[Function]

Get a fingerprint template of user information by string form.

[XML protocols]

Request Xml:

```
<GetUserTemplate>
        <ArgComKey xsi:type="xsd:integer”>ComKey</ArgComKey>
        <Arg>
           <PIN xsi:type="xsd:integer”>Job Number</PIN>
           <FingerID xsi:type="xsd:integer”>Finger Number</FingerID>
       </Arg>
```

```
    </GetUserTemplate>
```

Response Xml:

```
  <GetUserTemplateResponse>
          <Row>
              <PIN>XXXXX</PIN>
              <FingerID>XX</FingerID>
              <Size>XXX</Size>
              <Valid>X</Valid>
              <Template>XXXXXXXXXXXXXXXX….</Template>
      </Row>
  </GetUserTemplateResponse>
```

[Parameters]

ComKey: communications Password

PIN: User ID (Registration).

FingerID: fingerprints of users.

Pin: PIN of registered users.

Finge ID:the number of registered fingerprint .

Size: fingerprint size.

Template: fingerprint information.

Valid: Take effect

[Return value]

Return the fingerprint information of the assigned users

4.1.2.2 Write in the fingerprint template of user’s information

[Function]

Write in user fingerprint template by a string form. means that the users(DwPin) corresponding fingerprint template(TmpData) of finger(dwFingerIndex), were uploaded to the connected machines.

[XML protocols]

Request Xml:

```
<GetUserTemplate>
        <ArgComKey xsi:type="xsd:integer">ComKey</ArgComKey>
        <Arg>
            <PIN>XXXXX</PIN>
            <FingerID>XX</FingerID>
            <Size>XXX</Size>
            <Valid>X</Valid>
            <Template>XXXXXXXXXXXXXXXX….</Template>
        </Arg>
</GetUserTemplate>
```

Response Xml:

```
<GetUserTemplateResponse>
        <Row>
            <Result>Succeed! </Result>
        </Row>
</GetUserTemplateResponse>
```

[Parameters]

ComKey: communications Password Return

Pin: PIN of registered users.

Finge ID:the number of registered fingerprint .

Size: fingerprint size.

Template: fingerprint information.

Valid: Take effect

[Return value]

If successful return to True, or else return to False.

4.1.2.3 Delete users fingerprint template information

[Function]

Delete fingerprint template of user’s information.

[XML protocols]

Request Xml:

```
<DeleteTemplate>
        <ArgComKey xsi:type="xsd:integer">ComKey</ArgComKey>
        <Arg>
```

```
                  <PIN xsi:type="xsd:integer">Job Number</PIN>
             </Arg>
        </DeleteTemplate>
```

Response Xml:

```
  <DeleteTemplateResponse>
         <Row>
            <Result>Succeed! </Result>
        </Row>
  </DeleteTemplateResponse>
```

[Parameters]

ComKey: communications Password

PIN: User ID (Registration).

[Return value]

if successful True, or else return to False.

4.1.2.4 Delete all fingerprint template of information users

[Function]

Delete all fingerprint template of user's information .

[XML protocols]

Request Xml:

```
    <ClearData>
            <ArgComKey xsi:type="xsd:integer">ComKey</ArgComKey>
            <Arg>
            <Value xsi:type="xsd:integer">2</Value>
            </Arg>
   </ClearData>
```

Response Xml:

```
  <ClearDataResponse>
         <Row>
            <Result>Succeed! </Result>
      </Row>
  </ClearData>
```

[Parameters]

ComKey: communications Password

Value: Value operation

[Return value]

if successful to True, or else return to False.

4.1.3 Records Management

These records mainly include attendance record; these data can only be uploaded and is unable to be downloaded, read each record from the attendance machine.

4.1.3.1 Get all information log

[Function]

Read out the attendance record from attendance machines.

[XML protocols]

Request Xml:

```
<GetAttLog>
        <ArgComKey xsi:type="xsd:integer”>ComKey</ArgComKey>
        <Arg>
            <PIN xsi:type="xsd:integer”>Job Number</PIN>
      </Arg>
  </GetAttLog>
```

Response Xml:

```
<GetAttLogResponse>
      <Row>
          <PIN>XXXXX</PIN>
          <DateTime >YYYY-MM-DD HH:MM:SS</DateTime>
          <Verified>X</Verified>
          <Status>X</Status>
          <WorkCode>XXXXX</WorkCode>
    </Row>
</GetAttLogResponse>
```

[Parameters]

ComKey: communications Password

PIN: User ID (Registration).

[Return value]

If successful, return log information, or return Null.

Pin: User ID (Registration).

DateTime: Date Time.

Verified: Authentication method.

Status: Attendance status.

WorkCode: work code.

4.1.3.2 Delete all log information

[Function]

Delete all log information.

[XML protocols]

Request Xml:

```
<ClearData>
        <ArgComKey xsi:type="xsd:integer”>ComKey</ArgComKey>
        <Arg>
        <Value xsi:type="xsd:integer”>1</Value>
        </Arg>
</ClearData>
```

Response Xml:

```
<ClearDataResponse>
      <Row>
         <Result>Succeed! </Result>
    </Row>
</ClearData>
```

[Parameters]

ComKey: communications Password

Value: Value operation

[Return value]

if successful , return True, or else return to False.

4.1.4 Data Management System

4.1.4.1 Update information

[Function]

Refresh data.

[XML protocols]

Request Xml:

```
<RefreshDB>
        <ArgComKey xsi:type="xsd:integer”>ComKey</ArgComKey>
</RefreshDB>
```

Response Xml:

```
<DeleteUserResponse>
        <Row>
            <Result>Succeed! </Result>
        </Row>
</DeleteUserResponse>
```

[Parameters]

ComKey: communications Password

[Return value]

If successful, return True, or else return to False.

Note:

It be used to call, after upload users information or fingerprints, which will enable the modifications to make play, the role play synchronization.

4.2 machine settings

4.2.1 Restart equipment

[Function]

Restart machines.

[XML protocols]

Request Xml:

```
<Restart>
        <ArgComKey xsi:type="xsd:integer”>ComKey</ArgComKey>
</Restart>
```

Response Xml:

```
<RestartResponse>
        <Row>
            <Result>Succeed! </Result>
        </Row>
</RestartResponse>
```

[Parameters]

ComKey: communications Password

[Return value]
if successful, return True, or else return to False.

4.2.2 Get machine parameters

[Function]

Get machine parameters.

[XML protocols]

Request Xml:

```
<GetOption>
        <ArgComKey xsi:type="xsd:integer”>ComKey</ArgComKey>
        <Arg>
            <Name xsi:type="xsd:string”>Option Item Name</Name>
   </Arg>
</GetOption>
```

Response Xml:

```
<GetOptionResponse>
     <Row>
        <Name>XXXX</Name>
        <Value>XXXX</Value>
    </Row>
</GetOptionResponse>
```

[Parameters]

ComKey: communications Password

Name: parameter name.

[Return value]

Return to the parameter's name and value.

Name: parameter name,

Value: parameter values.

4.2.3 Get machines \ terminal time

[Function]

Get time of the machine \ terminals.

[XML protocols]

Request Xml:

```
<GetDate>
        <ArgComKey xsi:type="xsd:integer">ComKey</ArgComKey>
</GetDate>
```

Response Xml:

```
<GetDateResponse>
        <Row>
            <Date>YYYY-MM-DD</Date>
            <Time>HH:MM:SS</Time>
        </Row>
</GetDateResponse>
```

[Parameters]

ComKey: communications Password

[Return value]

Return time of the machine \ terminals

Date: Date

Time: time.

4.2.4 Machine parameter settings

[Function]

Machine Parameter settings .

[XML protocols]

Request Xml:

```
<SetOption>
        <ArgComKey xsi:type="xsd:integer">ComKey</ArgComKey>
        <Arg>
         <Name xsi:type="xsd:string">Item Name</Name>
         <Value xsi:type="xsd:string">Item Value</Value>
    </Arg>
</SetOption>
```

Response Xml:

```
<SetOptionResponse>
        <Row>
            <Result>Succeed! </Result>
        </Row>
</SetOptionResponse>
```

[Parameters]

ComKey: communications Password

Name: parameter name,

Value: parameter values.

[Return value]

If successful return True, or else return to False.

4.2.5 Set machines \ terminal time

[Function]

Set up the machine \ terminals time.

[XML protocols]

Request Xml:

```
<SetDate>
        <ArgComKey xsi:type="xsd:integer”>ComKey</ArgComKey>
        <Arg>
       <Date xsi:type="xsd:string”>YYYY-MM-DD</Date>
       <Time xsi:type="xsd:string”>HH:MM:SS</Time>
   </Arg>
</SetDate>
```

Response Xml:

```
<SetDateResponse>
      <Row>
         <Result> Succeed! </Result>
   </Row>
</SetDateResponse>
```

[Parameters]

ComKey: communications Password

Date: Date

Time: time.

[Return value]

if successful , return True, or else return to False.

5. FAQ

5.1 How to create users in machine on-line

use the function (SetuserInfo) to write users relevant records into machines, such as the registration numbers, passwords, names and other information

5.2 How to get all users information

GetUserInfo function can be used to get the user's information. If need the fingerprint template to be data GetUserTemplate function, the string type fingerprint template is available.

5.3 connected with machines

In the process of connecting the machine can be regarded as an independent PC, to connect. But pay attention to the machine's IP address and connected