App Note

VoxStack GSM Gateway API

Rev: 2.0

Date: On July 7, 2014

From: OpenVox support group

Contact Info: support@openvox.com.cn

Contents

OpenVox GSM Gateway HTTP Interface

OpenVox gateway supply plentiful external interfaces to meet clients' diversified needs about SMS service. OpenVox has been committed to better meet the needs of customers and create a convenient and flexible customer experience for use. The AMI and API interface consist of the whole and plentiful SMS system and interface type of the OpenVox GSM gateway for now.

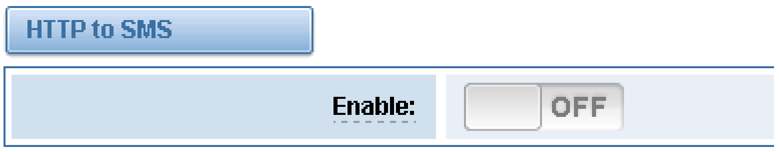
The AMI interface has been using early. The API interface is just completed.

The HTTP interface of OpenVox SMS is developed to be sample to use and meet customer's needs for the gateway compatibility with their SMS platform. We believe that you can spend less time and price to build your own SMS platform via using our HTTP interface. Absolutely, if you already have a stable and fully functional SMS platform, we also believe that you can make the SMS function of OpenVox GSM gateway compatible with your SMS platform via making a few changes, even no need to change.

HTTP to SMS Configuration

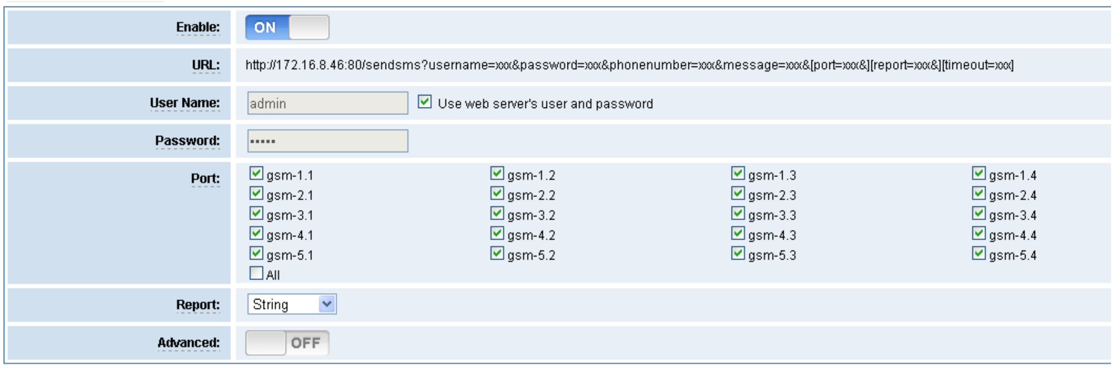
Sending SMS:

Step 1: Configure the HTTP SMS sending Interface on the OpenVox Gateway website. Enter the menu: SMS→SMS Settings→HTTP to SMS, as shown in the figure below.



HTTP to SMS	
Enable:	OFF

Step 2: Enable the HTTP to SMS function as below:



Enable:	ON			
URL:	http://172.16.8.46:80/sendsms?username=xxx&password=xxx&phonenumber=xxx&message=xxx&[port=xxx&][report=xxx&][timeout=xxx]			
User Name:	admin ✓ Use web server's user and password			
Password:	•••••			
Port:	✓ gsm-1.1	✓ gsm-1.2	✓ gsm-1.3	✓ gsm-1.4
	✓ gsm-2.1	✓ gsm-2.2	✓ gsm-2.3	✓ gsm-2.4
	✓ gsm-3.1	✓ gsm-3.2	✓ gsm-3.3	✓ gsm-3.4
	✓ gsm-4.1	✓ gsm-4.2	✓ gsm-4.3	✓ gsm-4.4
	✓ gsm-5.1	✓ gsm-5.2	✓ gsm-5.3	✓ gsm-5.4
	☐ All			
Report:	String			
Advanced:	OFF			

Enable: Enable the HTTP to SMS function, default is closed.

URL: Sending the SMS url, the parameters' introduction are as below:

Username: The user name for certification. You can define the username or just use the one for WEB login.

Password: The secret for certification. You can define the password or just use the one for WEB login.

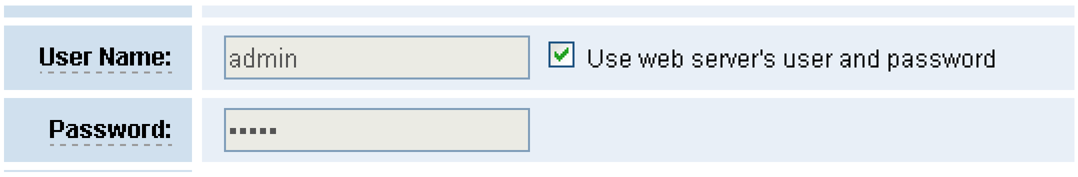
Phone number: The destination to send the SMS, usually the target cellphone number.

Message: The SMS context

Port: Optional parameter, need to specify which port to send, such as: gsm-1.2

Report: Optional parameter, to send the type of SMS report, such as: report=String

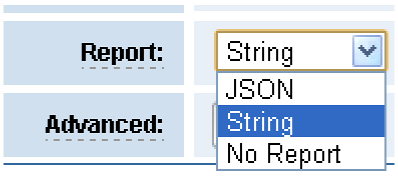
Timeout: Timeout, milliseconds, such as: timeout=5000



User Name:	admin ☑ Use web server's user and password
Password:	•••••

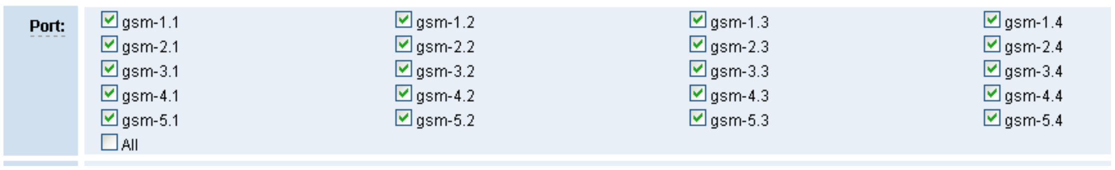
User Name: Custom user name, if u select the "Use web server's user and password", then use the WEB login username and secret.

Password: Custom secret, if select the "Use web server's user and password", then use the WEB login username and secret.



Report:	String
Advanced:	

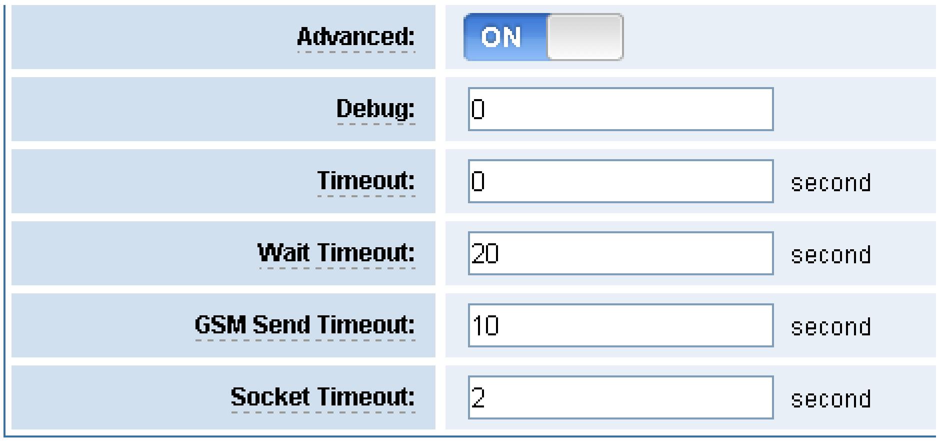
Report: The type of news report, supporting 2 kinds of news report format, String and JSON. If you choose "No Report", it means you don't need the news report.



Port:	☑ gsm-1.1	☑ gsm-1.2	☑ gsm-1.3	☑ gsm-1.4
	☑ gsm-2.1	☑ gsm-2.2	☑ gsm-2.3	☑ gsm-2.4
	☑ gsm-3.1	☑ gsm-3.2	☑ gsm-3.3	☑ gsm-3.4
	☑ gsm-4.1	☑ gsm-4.2	☑ gsm-4.3	☑ gsm-4.4
	☑ gsm-5.1	☑ gsm-5.2	☑ gsm-5.3	☑ gsm-5.4
	☐ All			

When the gateway receive a HTTP request to send SMS, it will choose one port from those you selected, for the descending order. For example, if sending SMS via gsm-1.1 port for the first time, then the second time it will choose the gsm-1.2, etc.

Advance options



Advanced:	ON
Debug:	0
Timeout:	0 second
Wait Timeout:	20 second
GSM Send Timeout:	10 second
Socket Timeout:	2 second

Debug: The level of debug messages (warning: this function is used to debug the error by technical support staff. Suggest keeping it in default.)

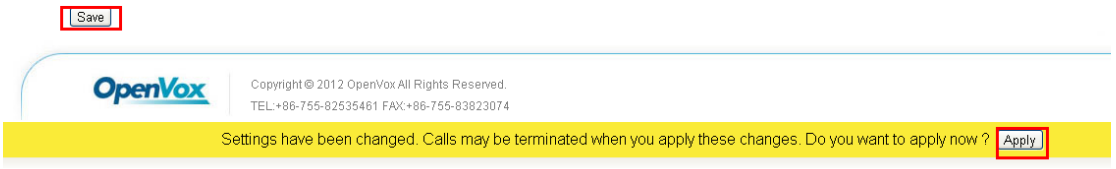
Timeout: Sending the timeout, default is 0, the unit is in seconds.

Wait Timeout: Waiting for the massage to report the timeout, default is 20 seconds, the unit is seconds.

GSM Send Timeout: GSM port send the timeout, default is 10 seconds, the unit is seconds.

Warning: Suggest to keep the Advance option in default or not to enable it.

(After you finish the configuration, please click the save button to save it and apply it as shown infigure below.)



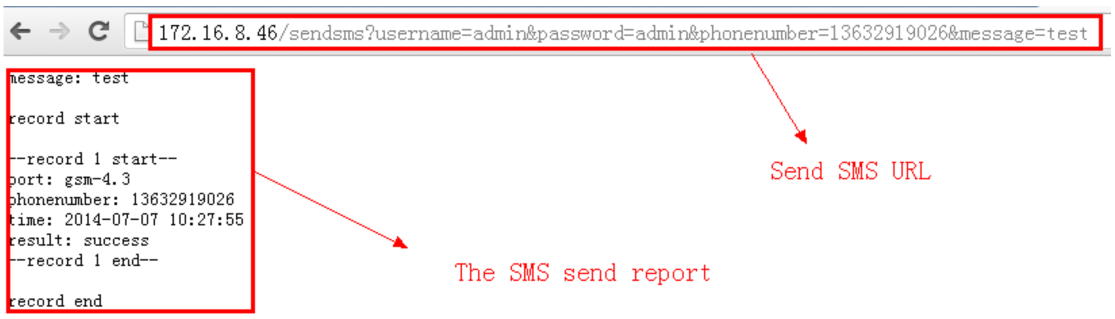
Save

Copyright © 2012 OpenVox All Rights Reserved.
TEL:+86-755-82535461 FAX:+86-755-83823074

Settings have been changed. Calls may be terminated when you apply these changes. Do you want to apply now ? Apply

Step 3: Use the HTTP interface to send SMS

Try to input a URL in the browser and click the ENTER key, the gateway will get the news report back to the current page, shown as figure below.



```
172.16.8.46/sendsms?username=admin&password=admin&phonenumber=13632919026&message=test
```

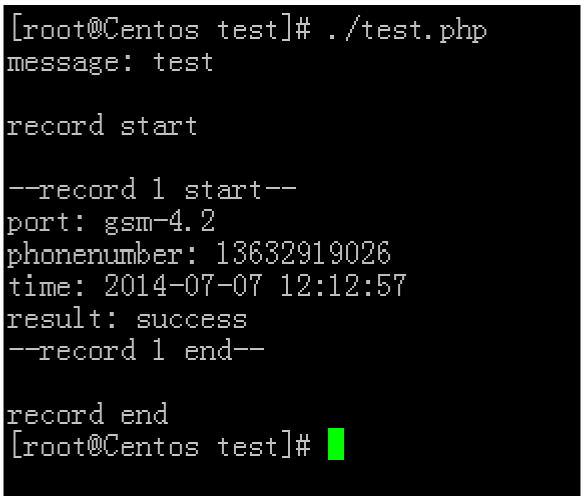
```
message: test

record start

--record 1 start--
port: gsm-4.3
phonenumber: 13632919026
time: 2014-07-07 10:27:55
result: success
--record 1 end--

record end
```

HTTP to SMS Samples

Use PHP to call the gateway HTTP interface, as follows:

```
#!/usr/bin/php -q
<?php
        function SendSMS()
        {
          // 1. Initialize an CURL session.
                  $ch = curl_init();

                  //2.Setting the request options, including specific URL.
                       curl_setopt($ch,CURLOPT_URL,"http://172.16.8.46:80/sendsms?
                       username=admin&password=admin&phonenumber=1363291902
                       6&message=test&port=gsm-4.2&report=String&timeout=5");

           // 3. Execute a CURL session and get the reply.
                      $response = curl_exec($ch);

           // 4. Release the Curl handle and close the CURL session.
                                curl_close($ch);
           }
           SendSMS();
 ?>
```

The results are as follows：



```
[root@Centos test]# ./test.php
message: test

record start

--record 1 start--
port: gsm-4.2
phonenumber: 13632919026
time: 2014-07-07 12:12:57
result: success
--record 1 end--

record end
[root@Centos test]#
```

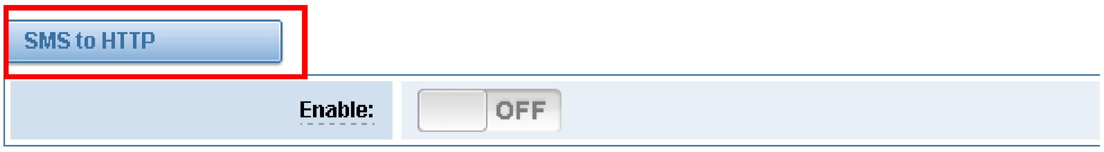
SMS to HTTP Configuration

Receiving SMS:

At present we delivered the SMS the gateway received to s specific URL link address in an active way.

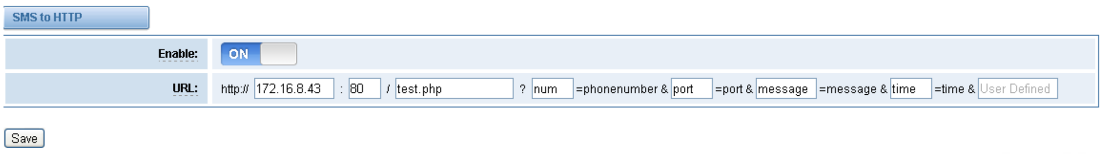
Step 1: Configure the HTTP interface to receive the SMS in OpenVox gateway.

Enter the menu: SMS→ SMS Settings→HTTP to SMS, shown as figure below:

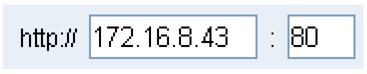


Enable:	OFF

Step 2: Enable the SMS to HTTP, shown as figure below：



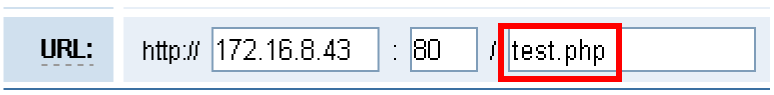
Enable:	ON
URL:	http:// 172.16.8.43 : 80 / test.php ? num =phonenumber & port =port & message =message & time =time & User Defined



http:// 172.16.8.43 : 80

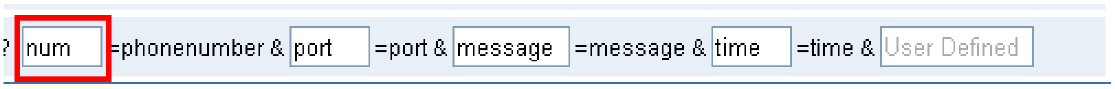
Hostname: the DNS host name or IP address of the server used to store resources. Sometimes, it also includes the user name and secret needed link to the server before the host name.

The integer is optional. When omit, use the default port. All kinds of protocols have default port number, such as HTTP port' default number is 80. Sometimes, for the sake of safety or others, we could redefine the port, using the non-standard port number. In this situation, the port number can not be omitted in the URL.



URL: http:// 172.16.8.43 : 80 / test.php

Separated by zero or more "/" symbol strings, commonly used to represent a directory or file address on the host.



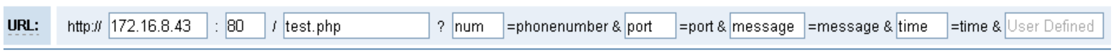
? num =phonenumber & port =port & message =message & time =time & User Defined

Used to deliver parameters to the dynamic web page
(Such as the pages made by CGI, ISAPI, PHP/JSP/ASP/ASP or NET technology, etc.)

Step 3: Use HTTP interface to receive SMS.

Warning: Before using this instance, please make sure you've installed and configured the web server correctly. This instance runs on the apache server.

The GSM gateway configuration is shown as follows:



URL: http:// 172.16.8.43 : 80 / test.php ? num =phonenumber & port =port & message =message & time =time & User Defined

SMS to HTTP Samples

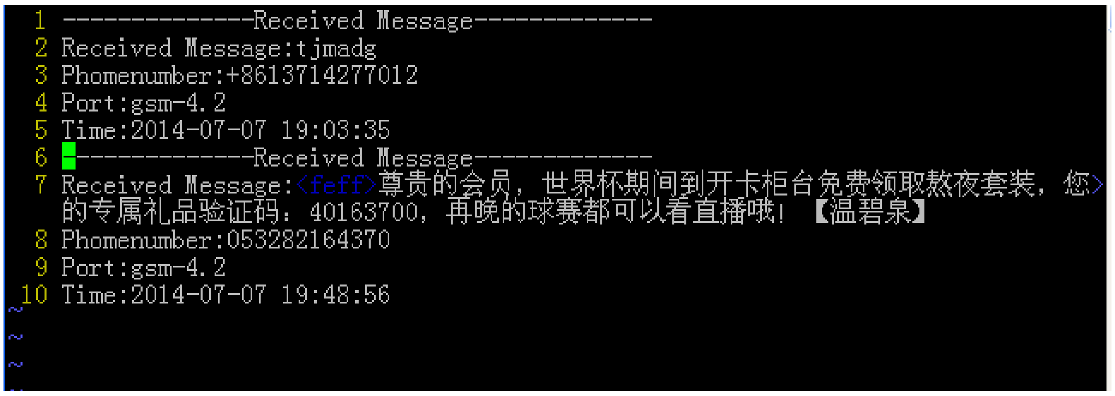
Edit the file test.php in the /var/www/html directory.

```
<?php
            $num=$_GET['num'];
            $port=$_GET['port'];
            $message=$_GET['message'];

            $Begin = "--------------Received Message-------------\n";
            $Message = "Received Message:".$message."\n";
            $MyPhonenumber = "Phomenumber:".$num."\n";
            $MyTime = "Time:".$time."\n";
            $MyPort = "Port:".$port."\n";

            $Time = date("Y-m-d H:i:s");
            $MyTime = "Time:".$Time;

            $MyStr = $Begin.$Message.$MyPhonenumber.$MyPort.$MyTime."\n";

            $file_pointer = fopen("/var/www/html/message.txt","a+");
            fwrite($file_pointer,$MyStr);
            fclose($file_pointer);

    ?>
```

Create the file message.txt in the /var/www/html directory.

Try to send a SMS to gateway and then open the message.txt. The result is as follows:

#vim message.txt



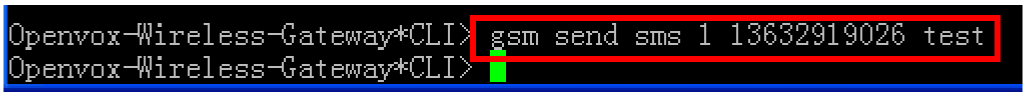
```
--------------Received Message-------------
Received Message:tjmadg
Phomenumber:+8613714277012
Port:gsm-4.2
Time:2014-07-07 19:03:35
--------------Received Message-------------
Received Message:<feff>尊贵的会员，世界杯期间到开卡柜台免费领取熬夜套装，您>
的专属礼品验证码：40163700，再晚的球赛都可以看直播哦！【温碧泉】
Phomenumber:053282164370
Port:gsm-4.2
Time:2014-07-07 19:48:56
~
~
~
```

From the result we can see that the gateway received two messages in 2014-07-07 both via gsm-4.2 port.

Supplement 1: Using Commands or AMI Interface to Send Long SMS

For now OpenVox gateway supports long SMS, no matter using WEB GUI or AMI/HTTP interface. The follows is instruction for sending long SMS via AMI interface.

Step 1: Try to send long SMS using the command (For more details, please learn our company's AMI documents), shown as figure below.



```
Openvox-Wireless-Gateway*CLI> gsm send sms 1 13632919026 test
Openvox-Wireless-Gateway*CLI>
```

Step 2: Sending long SMS.

Warning: To send long SMS (more than 160 bytes) via AMI interface, you need to break up your text messages (article points to send), and then to send it! That can ensure the long SMS you have sent for a whole article!

Sending command:

gsm send sync csms <span> <destination> <message> <flag> <smscount> <smssequence> <timeout>

span: the port used to send SMS, such as gsm-1.1 means to send long SMS using the first port

destination: the cellphone number to receive the SMS

message: the SMS context

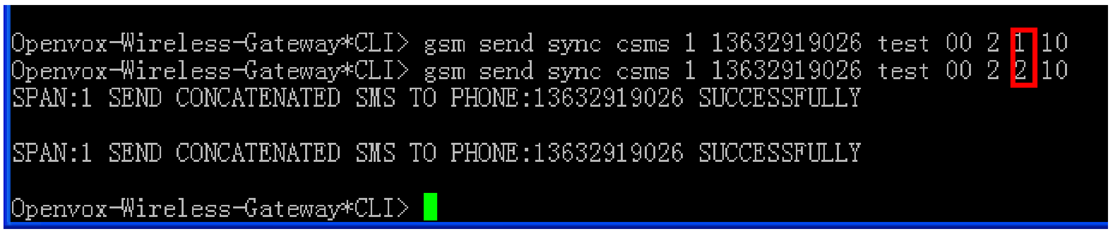
flag: the identity of the letter, for example, when you need to send a letter more than 160 bytes divided into two to send, you must ensure that these two orders have the same identity, scope of identity is from 00 to ff

smscount: the number of orders to split the letter, such as you need to divide the letter into 4 parts to send, then you should fill in 4

Smssequence: the queue to send, for example, you need to divide the letter into 2 parts to send, then the smssequence of the first order should be 1, the second order's should be 2, etc.

timeout: timeout to send, milliseconds

instance:



```
Openvox-Wireless-Gateway*CLI> gsm send sync csms 1 13632919026 test 00 2 1 10
Openvox-Wireless-Gateway*CLI> gsm send sync csms 1 13632919026 test 00 2 2 10
SPAN:1 SEND CONCATENATED SMS TO PHONE:13632919026 SUCCESSFULLY

SPAN:1 SEND CONCATENATED SMS TO PHONE:13632919026 SUCCESSFULLY

Openvox-Wireless-Gateway*CLI>
```

Instruction: In this instance dividing the letter of 8 bytes into 2 parts to send, firstly send a "test", secondly send other "test". But you only receive one letter and its context is "testtest" .

Supplement 2: Samples for Sending Long SMS via AMI Interface

Warning: to meet clients' diversified needs, this instance is written by C language.

Runtime environment: linux

Compile command: gcc sendsms.c –o sendsms

Runtime command: ./sendsms

The complete code is as follows.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>

//Login
void login_fun(int sock_fd)
{
    char username[20];
    char secret[20];
    int login_len=0;
    int secret_len=0;
    int res = 0;
    char receive_buf[4096];
    char login_buf[40];

    memset(username,'\0',20);
    memset(secret,'\0',20);
    printf("please input your username\n");
    scanf("%s",username);
    fflush(stdin);
    printf("please input your secret\n");
```

```
        scanf("%s",secret);
        fflush(stdin);

        memset(login_buf,'\0',40);
        sprintf(login_buf,"action:Login\r\nusername:%s\r\nsecret:%s\r\n\r\n",username,
secret);

        login_len = strlen(login_buf);

        if(res = write(sock_fd,login_buf,login_len) == login_len)
        {
                sleep(1);
                memset(receive_buf,'\0',4096);
                if(res = read(sock_fd,receive_buf,sizeof(receive_buf))<0)
                {
                        perror("login failed\n");
                        return;
                }
                printf("%s\n",receive_buf);
                if(NULL != strstr(receive_buf,"Authentication accepted"))
                {
                        printf("login success\n");
                }
        }
}



void sendsms_fun(int sock_fd)
{
        char send_buf[4096];
        char span_num[3];
        char destination[12];
        char message[2048];
        int res;
        int send_len;
        char receive_buf[4096];

        int SMS_Lengh = 160;
        int Message_Lengh = 0;
        int Divided_Lengh =152;
        char segmentation_message[153];
        int temp_count = 0;
        int smscount = 0;
        int smssequence = 1;
        int i = 0;
        char flag[3] = "00";
        int timeout = 10;

        memset(send_buf,'\0',4096);
        memset(destination,'\0',11);
        memset(message,'\0',2048);
        memset(span_num,'\0',3);

        printf("please input the span you want used\n");
        scanf("%s",span_num);
        fflush(stdin);
        printf("please input the destination num you want send\n");
        scanf("%s",destination);
        fflush(stdin);
        printf("Please input the message you want send\n");
        scanf("%s",message);
        fflush(stdin);

        Message_Lengh = strlen(message);
        printf("Input Message Lengh is:%d\n",Message_Lengh);
        if(Message_Lengh > SMS_Lengh)              //Send Long SMS
        {
                    if(Message_Lengh%Divided_Lengh == 0)
                        {
                                smscount = Message_Lengh/Divided_Lengh;
                        }
                        else
                        {
                                smscount = Message_Lengh/Divided_Lengh+1;
                        }
                        printf("The SMS will be divided into %d part send\n",smscount);
                        while(temp_count!=smscount)
                        {
                                for(i = 0;i<Divided_Lengh;i++)
                                {

                                    segmentation_message[i]=message[temp_count*Divide
                                    d_Lengh+i];
                                }

                                printf("The %d part is:\n",temp_count+1);
                                printf("%s\n",segmentation_message);
```

```
                                        sprintf(send_buf,"action:Command\r\ncommand:GSM send
sync
csms %s %s %s %s %d %d %d\r\n\r\n",span_num,destination,segmentation_message
,flag,smscount,smssequence+temp_count,timeout );
                                        printf("send_buf:%s\n",send_buf);
                                         send_len = strlen(send_buf);
                                         memset(receive_buf,'\0',4096);
                                    printf("%s\n",send_buf);
                                     if(res = write(sock_fd,send_buf,send_len) == send_len)
                                     {
                                         sleep(1);
                                        if(res = read(sock_fd,receive_buf,sizeof(receive_buf))<0)
                                        {
                                                  perror("send sms error\n");
                                                 return;
                                        }
                                        printf("%s\n",receive_buf);
                                        if(NULL != strstr(receive_buf,"Response: Follows"))
                                        {
                                                  printf("send SMS success\n");
                                        }
                                     }
                                      temp_count++;
                               }

    }
    else
    {
        sprintf(send_buf,"action:Command\r\ncommand:GSM send
sms %s %s %s\r\n\r\n",span_num,destination,message);
        send_len = strlen(send_buf);

        memset(receive_buf,'\0',4096);
        printf("%s\n",send_buf);
        if(res = write(sock_fd,send_buf,send_len) == send_len)
        {
            sleep(1);
            if(res = read(sock_fd,receive_buf,sizeof(receive_buf))<0)
            {
                perror("send sms error\n");
                return;
            }
            printf("%s\n",receive_buf);
            if(NULL != strstr(receive_buf,"Response: Follows"))
```

```
                {
                        printf("send SMS success\n");
                }
        }
    }
}

int main(void)
{
    int client_socket;
    struct sockaddr_in client_addr;
    char ServerIp[20];

    memset(ServerIp,'\0',20);
    printf("Please input your server ip address\n");
    scanf("%s",ServerIp);
    fflush(stdin);

    client_socket = socket(AF_INET,SOCK_STREAM,0);
    if(client_socket < 0)
    {
        perror("create socket error\n");
        return -1;
    }
    client_addr.sin_family  = AF_INET;
    client_addr.sin_port      = htons(5038);
    client_addr.sin_addr.s_addr  = inet_addr(ServerIp);

    if(connect(client_socket,(struct sockaddr *)&client_addr,sizeof(client_addr))<0)
    {
        perror("connect error\n");
        return -1;
    }
    else
    {
        printf("connect to %s success\n",ServerIp);
    }
    //login
    login_fun(client_socket);
    //send SMS
    sendsms_fun(client_socket);
    return 0;
}
```