Tornado 短信网关接口说明
（SOMP2-2.6.54）

1. CMPP2.0 协议

支持 CMPP2.x 协议，参考移动 CMPP2.0 协议说明
这里举一个使用 python 发送的例子（中文环境）：

```
#!/usr/bin/python
#coding:gb18030

import sys,socket,struct,time,hashlib,binascii

host = "222.33.44.111";
port = 7891
username = "test01";
password = "123456";
destAddr = "13612341234";
message = "你好,短信测试【测试】";
shortcode = "1234";
module = "TEST";

#1.create socket and connect
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM);
s.connect((host, port));

#2.bind
t = time.strftime("%m%d%H%M%S", time.gmtime());
m = hashlib.md5();
m.update( username );
m.update( "\x00\x00\x00\x00\x00\x00\x00\x00\x00" );
m.update( password );
m.update( t );
bind = struct.pack("!LLL6s16sBi", 39, 1, 1, username, m.digest(), 0x20, int(t));
s.sendall(bind);
bind_resp = s.recv(30);
status, = struct.unpack("!12xB17x", bind_resp);
print "bind: status=%d" % status
if status != 0:
    sys.exit();
```

```
#3.submit
i=len(message);
submit = struct.pack("!LLL8xBB2x10s24xB6s8s34x21sB21sB%ds8x" % i, 159+i, 4, 2,
    1, 1, module, 15, username, "01000000", shortcode, 1, destAddr, i, message);
s.sendall(submit);
submit_resp = s.recv(21);
msg_id,result = struct.unpack("!12x8sB", submit_resp);
print "submit: result=%d, msg_id=%s" %(result, binascii.hexlify(msg_id))

#4.terminate
term = struct.pack("!LLL", 12, 2, 3);
s.sendall(term);
term_resp = s.recv(12);

#5.close
s.close();
```

2. CMPP3.0 协议

支持 CMPP3.0 协议，参考移动 cmpp3.0 说明

3. CNGP 协议

支持 CNGP2.0 协议，参考原网通 CNGP 协议

4. EMPP 协议

支持 EMPPv2.0 协议，参考上海移动企信通 EMPP 协议
这里列举一个使用 PHP 发送的例子（中文环境）：

```
<?php
date_default_timezone_set("PRC");
$message = "你好,企信通测试【测试】";
$len = strlen($message);
$destaddr = "13622228888";

$shortcode = "10657001022776";
$password = "12345678";
```

```
// 1.create
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
$connection = socket_connect($socket, "211.136.163.68", 9981);

// 2.connect
$accountId = sprintf("%-'\x0021s", $shortcode);
$timestamp = date("mdHis");
$AuthenticatorSource=md5($shortcode
                ."\x00\x00\x00\x00\x00\x00\x00\x00\x00"
                .$password
                .$timestamp, true);
$connect = "\x00\x00\x00\x36\x00\x00\x00\x01\x00\x00\x00\x01"
                .$accountId
                .$AuthenticatorSource
                ."\x20"
                .pack("N", intval($timestamp) );
socket_write($socket, $connect);
$connect_resp = socket_read($socket, 256, PHP_BINARY_READ);

// 3.send message
$submit = "\x00\x00".chr(($len+215)>>8).chr(($len+215)&0xff)
          ."\x00\x00\x00\x04\x00\x00\x00\x02"
          ."\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
          ."\x01\x01\x00\x0F"
          ."\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
          ."\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
          ."\x00\x00\x00\x01"
          .$destaddr
          ."\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
          ."\x00\x00\x00\x00\x00"
          .chr($len)
          .$message
          .$accountId
          .$accountId
          .substr($accountId, 0, 10)
          ."\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
          ."\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
          ."\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
          ."\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
          ."\x00\x00";

socket_write($socket, $submit);
$submit_resp = socket_read($socket, 256, PHP_BINARY_READ);
```

```
// 4.disconnect
$terminate = "\x00\x00\x00\x0C\x00\x00\x00\x02\x00\x00\x00\x03";
socket_write($socket, $terminate);
$terminate_resp = socket_read($socket, 256, PHP_BINARY_READ);

// 5.close
socket_close($socket);
?>
```

5. SGIP 协议

支持 SGIP1.2 协议，参考联通 SGIP 协议

6. SMGP 协议

支持 SMGP1.3x、SMGP3.0 协议，参考电信 SMGP 协议

7. SMPP 协议

支持 SMPP3.3/3./5.0 协议，参考短信协议 smppv3.3/3.4/5.0

8. SOMP 协议

本平台特有协议。并提供 java 开发包和相关文档；使用该协议可以有效降低带宽，获得更大的发送速度。

9. HTTP 协议

目前支持 GET 方法。以下各参数未经说明不存在时,integer 作默认值 0 处理，string 则当作 null 处理。一般 integer(1)等同 byte, integer(4) 等同 int, integer(8)等同 long
发送短信入口 http://<host>:<port>/mt
参数说明如下：

参数	类型	说明	参考
dc	integer(1)	0 表示英文，8 表示 UCS2，15 表示中文	12.1

ec	integer(1)	一般不用	12.2
sm	string	默认 HEX 编码之消息内容；客户可以指定形式	12.3
pi	integer(1)	一般不用	12.4
da	string	手机号	12.5
sa	string	扩展码，必须以账号设定的开头；可以不填写	12.6
ld	string	linkID，目前不用	12.7
ex	integer(8)	外部编码，长整型	12.8
rd	integer(1)	是否需要状态报告	12.9
un	string	用户名	12.10
pw	string	密码，**切勿直接使用，注意安全**	12.11
st	string	定时发送时间；可不填	12.12
mu	string(32)	模块名，一般不用	12.13
pr	integer(1)	优先级	12.14
vp	string	有效期，可不填	12.15
rf	integer(1)	控制返回格式	12.16
ts	string	**时间标记，用于验证,格式 yyyyMMddHHmmss**	12.26
tf	integer(1)	短信内容的传输编码，默认为 0 表示 HEX 格式	12.28
rl	integer(4)		

颜色标出为必须填写

发送示例,消息内容为“ABC”(414243 为十六进制表示)，需要状态报告(rd=1):

http://10.10.10.5:8088/mt?un=star&pw=123456&da=13612345678&sm=414243&dc=15&rd=1

或者这样发送，内容为“ABC”,设 置 tf=2；需使用 URLEncoder：

http://10.10.10.5:8088/mt?un=star&pw=123456&da=13612345678&sm=ABC&dc=15&tf=2

不同号码不同内容群发

da	string	空（或不存在），此时默认分割符“|”（半角）；或者分隔符字符串，长度 1~4，**必须半角**，比如“$$”，但不可以为“#”	12.5
sm	string	默认 HEX 编码之消息内容；编码前格式为“手机号#外部编号#内容|手机号#外部编号#内容|.....”，其中的分割符“|”可以通过参数 da 来指定。分割符后面不可以有空格。如果没有指明外部编号的，则使用参数“ex”的设置。“外部编号”定义同参数“ex”定义。 举例: “13812345678#1234#测试短信” 详细请查看示例代码。	12.3

使用该方式发送时，请务必确保手机号正确，内容合法。

发送结果,字段含义如下:

参数	类型	说明	参考
r	integer(4)	错误码，不存在时认为是 0，即没有错误。	12.17
id	string	消息编号，成功时返回，失败时该字段可省略	12.18

发送结果返回举例:

当 **rf=0** 时，此为**默认值**，各字段以&分割,	
成功	id=<消息编号> 或者 r=0&id=<消息编号>
失败	r=<错误码>
当 **rf=2** 时，即 json 格式。**可能会出现未定义字段，忽略即可**。缺失字段则按默认处理。	
成功	{"id":"<消息编号>"}
失败	{"r":"9103"}

想发送中文，知道 UTF8 编码的，可以尝试使用 tf=3,dc=15 的设置。具体解释请参考 tf 说明。

接收短信入口 http://<host>:<port>/mo
接收状态报告和上行信息都在该地址

参数	类型	说明	参考
un	string	用户名	12.10
pw	string	密码	12.11
fs	integer(4)	返回大小设定	12.27
ts	string	**时间标记，用于验证,格式 yyyyMMddHHmmss**	12.26
rf	integer(1)	控制返回格式	12.16
tf	integer(10)	短信内容的传输编码，默认为 0 表示 HEX 格式	12.28

应答消息以多行返回消息(”\r\n”分割)每行一个。最后一个行分割可能不存在。
行间参数用&分割

参数	类型	上行消息说明	状态报告说明	参考
op	string	mo	dr	12.19
dc	integer(1)	消息类型		12.1
pi	integer(1)	cmpp 之 tp_pid		12.4
ec	integer(1)	cmpp 之 tp_udhi		12.2
sa	string	手机号		12.6
da	string	扩展码		12.5
mu	string	模块名		12.13
sm	string	HEX 编码之消息内容		12.3
id	string		消息编号	12.18
ex	integer(8)		外部编码	12.8
su	string		状态说明	12.20
sd	string	接收时间	提交时间	12.21
dd	string		完成时间	12.22
rp	integer(4)		错误码	12.23
bi	integer(4)		拆分编号	12.24
di	integer(4)		群发时号码的位置	12.25

除 op 之外,状态报告和 mo 的各个参数未必都存在，特别是兰色标出的。

客户也可根据需要申请主动推送之 http 接口，即由我方服务器向客户方服务器主动发送 http 请求，每次一个数据；参数如上说明,不再赘述。

这是推荐方式，可以保证上行和状态报告的及时性。

关于群发时装态报告返回的编号 id，一般为提交时返回的编号，客户可以以该 id 和手机号为规则进行对应；表中<bi><di>分别和内容拆分和号码拆分有关，如果存在都是从 1 开始。如果账号设置了需要连续编号，则对于群发 n 个号码，状态报告返回的编号为 id+0 到 id+n-1。不建议群发。

接收上行和状态报告举例
http://10.10.10.5:8088/mo?un=star&pw=123456
如需返回 xml 格式的响应
http://10.10.10.5:8088/mt?un=star&pw=123456&rf=1

举例 1：密未经处理，使用 HEX 发送式（java+httpclient4+codec）：

```
String mobile = "13612345678"; // 手机号
String username = "test"; // 用户名
String password = "123456"; // 密码
int dataCoding = 8; // UNICODE 编码（UTF-16BE）

// 使用 Hex 编码内容
String message = "这是一条测试短信";
String hex = Hex.encodeHexString( message.getBytes("UTF-16BE") );

StringBuilder sb = new StringBuilder();
sb.append("http://localhost:7890/mt?")
    .append("un=").append( username )
    .append("&pw=").append(URLEncoder.encode(password,"utf8") )
    .append("&da=").append( mobile )
    .append("&dc=").append( dataCoding )
    .append("&sm=").append( hex );

String req = sb.toString();
System.out.println("request: " + req);
String result = Request.Get( req ).connectTimeout(60000).socketTimeout(60000)
    .execute().returnContent().asString();
System.out.println( "response: " + result );
```

发送成功的应答如下：

```
id=142551080979073768
```

举例 2：密码经过 MD5 处理,使用 URLEncode+UTF8（java+httpclient4）：

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
```

```
String timestamp = sdf.format(new Date() );

String mobile = "13612345678"; // 手机号
String shortCode = "8888"; // 扩展码
String username = "test"; // 用户名
String password = "123456"; // 密码
long externalId = 0x123456789L; // 自定义消息编码，可以忽略
int dataCoding = 8; // UNICODE 编码
int transferEncoding = 3; // URLEncode+UTF8
int responseFormat = 2; // 返回格式为 Json 格式
String message = "这是一条测试短信,返回 Json";

// 计算密码摘要
SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
String timestamp = sdf.format(new Date() );
MessageDigest md5 = MessageDigest.getInstance("MD5");
md5.update( username.getBytes("utf8") );
md5.update( password.getBytes("utf8") );
md5.update( timestamp.getBytes("utf8") );
md5.update( message.getBytes("utf8") );
password = Base64.encodeBase64String( md5.digest() );

StringBuilder sb = new StringBuilder();
sb.append("http://localhost:7890/mt?")
      .append("un=").append( username )
      .append("&pw=").append( URLEncoder.encode(password,"utf8") )
      .append("&ts=").append( timestamp )
      .append("&da=").append( mobile )
      .append("&sa=").append( shortCode )
      .append("&ex=").append( externalId )
      .append("&dc=").append( dataCoding )
      .append("&tf=").append( transferEncoding )
      .append("&rf=").append( responseFormat )
      .append("&sm=").append( URLEncoder.encode(message, "utf8") );

String req = sb.toString();
System.out.println("request: " + req);
String result = Request.Get( req ).connectTimeout(60000).socketTimeout(60000)
      .execute().returnContent().asString();
System.out.println( "response: " + result );
```

返回格式参考（Json 格式）：

```
{"success": true, "id": "142540128277943843"}
```

举例 3：密码为明码,使用 URLEncode+UTF8（java+httpclient4）：

```
String mobile = "13612345678"; // 手机号
String shortCode = "8888"; // 扩展码
String username = "test"; // 用户名
String password = "123456"; // 密码
long externalId = 0x123456789L; // 自定义消息编码，可以忽略
int dataCoding = 8; // UNICODE 编码
int transferEncoding = 3; // URLEncode+UTF8
int responseFormat = 1; // 返回格式为 xml 格式
String message = "这是一条测试短信，返回 XML";

StringBuilder sb = new StringBuilder();
sb.append("http://localhost:7890/mt?")
        .append("un=").append( username )
        .append("&pw=").append(URLEncoder.encode(password,"utf8") )
        .append("&da=").append( mobile )
        .append("&sa=").append( shortCode )
        .append("&ex=").append( externalId )
        .append("&dc=").append( dataCoding )
        .append("&tf=").append( transferEncoding )
        .append("&rf=").append( responseFormat )
        .append("&sm=").append( URLEncoder.encode(message,    "utf8") );

String req = sb.toString();
System.out.println("request: " + req);
String result = Request.Get( req ).connectTimeout(60000).socketTimeout(60000)
        .execute().returnContent().asString();
System.out.println( "response: " + result );
```

返回格式参考（成功的 XML）：

```
<?xml version="1.0"?>
<id>1425402614219300 21</id>
```

举例 4：获得发送结果，密码为明码：

```
String username = "test"; // 用户名
String password = "123456"; // 密码
int fetchSize = 100;// 每次返回的最大条数

StringBuilder sb = new StringBuilder();
sb.append("http://localhost:7890/mo?")
        .append("un=").append( username )
        .append("&pw=").append(URLEncoder.encode(password,"utf8"))
        .append("&fs=").append( fetchSize )
```

```
        .append("&rf=").append( 2 ); // 控制返回格式，可不加

    String req = sb.toString();
    System.out.println("request: " + req);
    String result = Request.Get( req ).connectTimeout(60000).socketTimeout(60000)
        .execute().returnContent().asString();
    System.out.println( "response: " + result );
```

举例 5：获得发送结果，密码经过 MD5 处理：

```
    String username = "test"; // 用户名
    String password = "123456"; // 密码
    int fetchSize = 100;// 每次返回的最大条数

    SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
    String timestamp = sdf.format(new Date() );
    MessageDigest md5 = MessageDigest.getInstance("MD5");
    md5.update( username.getBytes("utf8") );
    md5.update( password.getBytes("utf8") );
    md5.update( timestamp.getBytes("utf8") );
    password = Base64.encodeBase64String( md5.digest() );

    StringBuilder sb = new StringBuilder();
    sb.append("http://localhost:7890/mo?")
        .append("un=").append( username )
        .append("&pw=").append(URLEncoder.encode(password,"utf8") )
        .append("&fs=").append( fetchSize )
        .append("&ts=").append( timestamp )
        .append("&rf=").append( 1 ); // 控制返回格式，可不加

    String req = sb.toString();
    System.out.println("request: " + req);
    String result = Request.Get( req ).connectTimeout(60000).socketTimeout(60000)
        .execute().returnContent().asString();
    System.out.println( "response: " + result );
```

举例 6：C#发送：

```
    string username = "test";
    string password = "123456"; // 如果有特殊字符，请使用 UrlEncode 处理
    string message = "短信发送测试【测试】";
    Encoding enc = Encoding.GetEncoding("UTF-16BE");
    message = BitConverter.ToString(enc.GetBytes(message)).Replace("-", "");
    string destAddr = "13600000000";
```

```
String url = "http://192.168.1.100:7891/mt?un=" + username
      + "&pw=" + password + "&da=" + destAddr + "&dc=8&sm=" + message;

WebClient client = new WebClient();
Stream stream = client.OpenRead(url);
StreamReader r = new StreamReader(stream);
string result = r.ReadToEnd();
stream.Close();
Console.WriteLine(result);
```

举例 7：中文不同内容短信群发的参数处理（java）：

```
String mobile = ""; // 参数 da：使用默认分割符|
int dataCoding = 15; // 参数 dc：中文编码
int transferEncoding = 3; // 参数 tf：URLEncode+UTF8
String message = "13622220001#11#测试短信 1|13622220002#12#测试短信 2";
....
.append("&sm=").append( URLEncoder.encode(message, "utf8") );
....
```

举例 8：UCS2 不同内容短信群发的参数处理（java）：

```
String mobile = "$$"; // 参数 da：自定义分割符$$
int dataCoding = 8; // 参数 dc：使用 UCS2 编码，支持国际语言
int transferEncoding = 2; // 参数 tf：URLEncode
String [] messages = {"第一条 Тестовое сообщение",
                      "第二条 Тестовое сообщение2"};
String[] mobiles = { "1381234001","1381234002" };
StringBuilder msg = new StringBuilder();
for(int i = 0; i < messages.length; i++) {
      if ( i > 0 ) msg.append( mobile );
      msg.append(mobiles[i] ).append("#");
      msg.append( new String( messages[i].getBytes("UTF-16BE"), "ISO8859_1") );
}
String message = msg.toString();
....
.append("&da=").append( URLEncoder.encode(mobile, "utf8") );
.append("&sm=").append( URLEncoder.encode(message, "ISO8859_1") );
....
```

10. WebService 接口

接口入口如下（POST 方式访问），IP 地址和端口另外确定
http://10.10.10.5:8088/wbs
获得接口描述文档
http://10.10.10.5:8088/wbs?wsdl

10.1. 发送消息 SubmitSM

由客户端发起，用于发送短信。详细定义请参考 service.wsdl 文档，以下为参数解释

参数	类型	说明	参考
username	xsd:string	用户名	12.10
password	xsd:string	密码，**切勿直接使用，注意安全**	12.11
timestamp	xsd:string	**时间标记，用于验证**	12.26
dataCoding	xsd:int	编码，15 为中文	12.1
esmClass	xsd:int	可不填	12.2
content	xsd:string	短信内容	12.3
protocolID	xsd:int	可不填	12.4
destAddr	xsd:string	手机号	12.5
sourceAddr	xsd:string	扩展码	12.6
linkID	xsd:string	可不填	12.7
externalID	xsd:long	客户端编号，可不填	12.8
registeredDelivery	xsd:int	是否需要状态报告	12.9
scheduleTime	xsd:string	定时时间	12.12
module	xsd:string	模块名	12.13
priority	xsd:int	优先级	12.14
validityPeriod	xsd:string	有效时间	12.15

发送消息的应答

参数	类型	说明	参考
result	xsd:int	结果	12.17
messageID	xsd:string	如果结果为 0，则返回消息编号	12.18

10.2. 接收消息 DeliverSM

由客户端发起，用于接收状报告和上行。详细定义请参考 service.wsdl 文档，以下为参数解释

参数	类型	说明	参考
username	xsd:string	用户名	12.10
password	xsd:string	密码	12.11

timestamp	xsd:string	时间标记	12.26
fetchSize	xsd:int	可以返回的数量	12.27

接收消息的应答

参数	类型	说明	参考
result	xsd:int	结果	12.17
deliverSM	tns:DataSM	上行信息和状态报告的数组	

其中 tns:DataSM 的类型说明如下：

参数	类型	说明	参考
operation	xsd:string	数据类型 mo/dr	12.17
dataCoding	xsd:int	短信内容编码	12.1
protocolID	xsd:int	短信协议	12.4
esmClass	xsd:int	编码类别	12.2
sourceAddr	xsd:string	手机号	12.6
destAddr	xsd:string	扩展码	12.5
module	xsd:string	模块名	12.13
content	xsd:string	短信内容	12.3
messageID	xsd:string	消息编号	12.18
externalID	xsd:long	外部编号	12.8
status	xsd:string	状态报告描述字符串	12.20
submitDate	xsd:string	提交时间	12.21
doneDate	xsd:string	完成时间	12.22
receipt	xsd:int	状态报告结果	12.23
blockIndex	xsd:int	拆分序号	12.24
destIndex	xsd:int	分割序号	12.25

10.3. 推送消息 DeliverSM

由服务器发起，用于投递状态报告和上行。详细定义请参考 receipt.wsdl 文档，以下为参数解释

参数	类型	说明	参考
deliverSM	tns:DataSM	上行信息和状态报告的数组	

tns:DataSM 说明参上节

11. 数据库接口

11.1. SQLSERVER2005

jdbc 连接字符串如下（IP 和端口请根据实际修改）：

```
jdbc:sqlserver://10.10.1.100:1433;databaseName=sms;integratedSecurity=false;encrypt=false;
```

驱动使用 sqljdbc4.jar

建立数据库连接：

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
conn =  DriverManager.getConnection(dbUrl, username, password);
```

消息发送（示例代码，请务必根据需要修改）：

```
// 发送一条消息
ptmt = conn.prepareCall("{ ?=call smsmt(?,?,?,?,?,?) }");

ptmt.registerOutParameter(1, Types.INTEGER);
ptmt.registerOutParameter(2, Types.BIGINT );

ptmt.setString(3, "13612345678"); // 手机号
ptmt.setNString(4, message); // 消息

// 以下三句可以倒序逐次省略，请同时修改 prepareCall 部分的问号数量
ptmt.setInt(5, 1); // 是否需要状态报告，默认 0
ptmt.setString(6, "8888"); // 扩展码，默认为账号设置
ptmt.setLong(7, externalId); // 客户自己设定的消息编号，默认 0

ptmt.execute();
int result = ptmt.getInt(1);
long msgId = ptmt.getLong(2);
if ( result == 0 ) {
    // 发送成功,msgId 有效
} else {
    // 发送失败,msgId 无效,切勿使用
}
```

消息接收和状态报告（示例代码，请务必根据需要修改，示例代码列出较多字段，请依据实际情况筛选）：

```
// 查询上行和状态报告
ptmt = conn.prepareCall("{ call smsmo(?) }");
ptmt.setInt(1, 100); // 每次返回的最大数量
rs = ptmt.executeQuery();
while ( rs.next() ) {
    String op = rs.getString("op");
    if ( "dr".equals(op) ) {
        String mobile =rs.getString("sa"); // 手机号码
        String shortCode = rs.getString("da"); // 扩展码
        long msgId = rs.getLong("id"); // 消息编号
```

```
                long externalId = rs.getLong("ex"); // 客户发送时自定义的消息编号
                String status = rs.getString("su"); // 状态字符串
                Timestamp submitDate = rs.getTimestamp("sd"); // 提交时间
                Timestamp doneDate = rs.getTimestamp("dd"); // 完成时间
                int receipt = rs.getInt("rp"); // 错误码
                System.out.println("收到状态报告: mobile=" + mobile + "; msgId=" + msgId
                        + "; externalId=" + externalId + "; status=" + status
                        + "; result: " + receipt);
            } else if ( "mo".equals(op) ) {
                int dataCoding = rs.getInt("dc"); // 消息编码
                int protocolId = rs.getInt("pi");
                int esmClass = rs.getInt("ec");
                String mobile = rs.getString("sa"); // 手机号码
                String shortCode = rs.getString("da"); // 扩展码
                String module = rs.getString("mu"); // 模块名
                String message = rs.getString("sm"); // 消息,注意当 esmClass 非零时，
                                                     // 这里是 Hex 编码
                System.out.println("收到上行信息: mobile=" + mobile
                        + "; shortCode=" + shortCode
                        + "; message=" + message);
            } else {
                System.out.println("unknown op: " + op);
            }
        }
        rs.close();
        ptmt.close();
```

以上字段可以参考 HTTP 接口的说明。

建议在一个连接上定时发起接收状态报告和上行，用来保持连接，否则服务器会将空闲链接关闭。

在发送消息时完整的参数顺序如下：

序号	类型	说明
1	integer(4)	返回结果,错误码
2	integer(8)	成功时为消息编号
3	string	手机号码
4	string	消息内容（当 esmClass 不是 0 时使用 Hex 编码）
5	integer(1)	是否需要状态报告
6	string	扩展码
7	integer(8)	客户自定义编号
8	integer(1)	esmClass
9	integer(1)	消息编码，默认 8
10	integer(1)	protocoID
11	string	模块名

12	integer(1)	优先级
13	timestamp	定时时间
14	timestamp	有效期

12. XMPP 协议

基本配置，域固定为 xmpp.org, 请另外设置服务器的端口，IP 地址

12.1. 使用 xmpp 客户端(chat 模式)发送接收短信

发送短信
目标为：smsc@xmpp.org
内容为：手机号：内容

接收短信
内容为：手机号：内容

接收状态报告：
内容为：手机号#状态

发送失败
内容为：ERROR:错误码

使用 smack-4.1.8 的可参考如下例子：

```
String domain = "xmpp.org";

XMPPTCPConnectionConfiguration conf = XMPPTCPConnectionConfiguration.builder()
                .setConnectTimeout(60000)
                .setHost("localhost")
                .setPort(5222)
                .setSendPresence(false)
                .setSecurityMode(SecurityMode.disabled)
                .setUsernameAndPassword("test", "123456")
                .setCompressionEnabled(false)
                .setServiceName(domain)
                .build();

  XMPPTCPConnection conn = new XMPPTCPConnection(conf);
  conn.connect();
  conn.login();
```

```
System.out.println("login ok");

ChatManager cm = ChatManager.getInstanceFor(conn);

Chat chat = cm.createChat("smsc@" + domain, new ChatMessageListener() {
    @Override
    public void processMessage(Chat chat, Message message) {
        System.out.println("Received message: " + message);
    }
});

chat.sendMessage("hello!");

System.out.println("send message ok");

System.out.println("wait 5000");
Thread.sleep(10000L);

conn.disconnect();
System.out.println("disconnect ok");
```

12.2. 使用 xmpp-api 发送接收短信

发送短信
目标为：手机号码@xmpp.org
thread：客户自定义短信编号，即 external_id
subject：扩展码
type：normal, 如果填写 chat 则进入 chat 模式(前一节说明)
内容为：短信内容

接收短信：
来源：手机号码@xmpp.org/mo
subject：扩展码
thread：服务端指定编号
内容为：短信内容

接收状态报告：
来源：手机号码@xmpp.org/dr
thread：下发时指定的 thread
subject：扩展码
内容为：状态

使用 smack-4.1.8 的可参考如下例子：

```
String domain = "xmpp.org";

XMPPTCPConnectionConfiguration conf = XMPPTCPConnectionConfiguration.builder()
                .setConnectTimeout(60000)
                .setHost("localhost")
                .setPort(5222)
                .setSendPresence(false)
                .setSecurityMode(SecurityMode.disabled)
                .setUsernameAndPassword("test", "123456")
                .setCompressionEnabled(false)
                .setServiceName(domain)
                .build();

 XMPPTCPConnection conn = new XMPPTCPConnection(conf);
 conn.connect();
 conn.login();
 System.out.println("login ok");

 conn.addSyncStanzaListener(new StanzaListener() {
                @Override
                public void processPacket(Stanza arg0) throws NotConnectedException {
                    System.out.println( "From: " + arg0.getFrom() );
                    System.out.println( "To: " + arg0.getTo() );
                    System.out.println( "StanzaId: " + arg0.getStanzaId() );
                    Message msg = (Message) arg0;
                    System.out.println( "subject: " + msg.getSubject() );
                    System.out.println( "body: " + msg.getBody() );
                    System.out.println( "thread: " + msg.getThread() );
                }
            },
            new StanzaFilter() {
                @Override
                public boolean accept(Stanza arg0) {
                    if ( arg0 instanceof Message ) {
                        return true;
                    }return false;
                }

 } );

 String thread = Long.toString( System.currentTimeMillis() );
 System.out.println("send message id: " + thread);
```

```
        Message msg = new Message("13812345678@" + domain, "hello!");
        msg.setSubject("5555");
        msg.setThread( thread );
        conn.sendStanza(msg);

        System.out.println("send message ok");

        System.out.println("wait 5000");
        Thread.sleep(10000L);

        conn.disconnect();
        System.out.println("disconnect ok");
```

13. CORBA 接口

IDL 定义文件如下：

```
module somp {

   // 0 表示忽略，毫秒，相对于 midnight, January 1, 1970 UTC.
   typedef long long LongTime;

   struct MD5WithTimeData {
      string username;
      LongTime timestamp;
      long long nonce; // 非负
      octet sign[16];
   };

   const long SecurityContextID = 983055;

   typedef long SecurityMethod;
   const long MD5WithTime = 1;

   union SecurityContext switch(SecurityMethod) {
      case MD5WithTime: MD5WithTimeData md5t;
   };

   interface sms {
      struct SubmitReq {
         string sourceAddr;
         string destAddr;
```

```
    sequence<octet> content;
    long registeredDelivery;
    long long externalID; // 非负
    long dataCoding;
    long esmClass;
    long protocolID;
    string moduleName;
    long priority;
    LongTime scheduleTime;
    LongTime validityPeriod;
};
struct SubmitRsp {
    long result;
    unsigned long long messageID;
};
SubmitRsp submit(in SubmitReq req);

struct QueryReq {
    long fetchSize;
};

struct DataSM_SmsMO {
    long dataCoding;
    long protocolID;
    long esmClass;
    string sourceAddr;
    string destAddr;
    string moduleName;
    sequence<octet> content;
    LongTime submitDate;
};

struct DataSM_SmsDR {
    string sourceAddr;
    string destAddr;
    string moduleName;
    long long messageID;
    long long externalID;
    string status;
    LongTime submitDate;
    LongTime doneDate;
    long receipt;
    long blockIndex;
    long destIndex;
```

```
    };

    typedef long OperationType;
    const long OpSmsMO = 1;
    const long OpSmsDR = 2;

    union DataSM switch ( OperationType ) {
    case OpSmsMO: DataSM_SmsMO smsmo;
    case OpSmsDR: DataSM_SmsDR smsdr;
    };

    struct QueryRsp {
        sequence<DataSM> results;
    };
    QueryRsp query(in QueryReq req);
  };
};
```

SecurityContextID 这个是用于认证的 ServiceContext 的编号，
SecurityContext 即为 ServiceContext 的数据部分结构。

MD5WithTimeData 中的 sign 计算如下(Java 代码)：

```
sign = DigestUtils.md5( username + nonce + password + timestamp );
```

nonce 需满足以下条件，假设本次使用的 timestamp 是 t1,nonce 是 n1；下次使用的是 t2,n2
则：**t1<t2 || (t1==t2 && n1<n2) 为真**

14. MS-RPC

RPC 接口，支持 NTLM 认证方式,IDL 文件定义如下

```
import "oaidl.idl";

[
uuid(8876B53A-A015-4B10-8FE6-CEB81943EDD3),
version(1.0),
pointer_default(unique)
]
interface ISmsApi
{
    typedef long long LONG64;

    typedef struct tagSUBMIT_REQ {
```

```
        char sourceAddr[24];
        UP_BYTE_BLOB destAddr;
        UP_BYTE_BLOB content;
        LONG64 externalID;
        long registeredDelivery;
        long dataCoding;
        long esmClass;
        long protocolID;
        char moduleName[12];
        long priority;
        LONG64 scheduleTime;
        LONG64 validityPeriod;
    } SUBMIT_REQ;


[id(1)] long Submit(
    [in] SUBMIT_REQ* pREQ,
    [out] LONG64 * pMessageID);


    typedef struct tagSMSMO {
        long dataCoding;
        long protocolID;
        long esmClass;
        char sourceAddr[24];
        char destAddr[24];
        char moduleName[12];
        UP_BYTE_BLOB content;
        LONG64 submitDate;
    } SMSMO;

    typedef struct tagSMSDR {
        char sourceAddr[24];
        char destAddr[24];
        LONG64 messageID;
        LONG64 externalID;
        long receipt;
        char status[8];
        LONG64 submitDate;
        LONG64 doneDate;
        long blockIndex;
        long destIndex;
    } SMSDR;
```

```
        typedef union tagDATA_SM switch( long opType) data {
                case 1: SMSMO smsmo;
                case 2: SMSDR smsdr;
        } DATA_SM;

[id(2)] long Query(
        [in] long nMax,
        [out] long * pActual,
        [out, size_is(nMax),length_is(*pActual) ] DATA_SM**pData);

[id(3)] LONG64 Balance();

}
```

认证方式设置参考如下：

```
        SEC_WINNT_AUTH_IDENTITY_W authid;
        authid.Domain = NULL;
        authid.DomainLength = 0;
        authid.User = reinterpret_cast<USHORT*>(pszName);
        authid.UserLength = wcslen(pszName);
        authid.Password = reinterpret_cast<USHORT*>(pszPassword);
        authid.PasswordLength = wcslen(pszPassword);
        authid.Flags = SEC_WINNT_AUTH_IDENTITY_UNICODE;

        status = RpcBindingSetAuthInfoEx(Binding, NULL,
                RPC_C_AUTHN_LEVEL_CONNECT,
                RPC_C_AUTHN_WINNT,
                &authid,
                0,
                NULL);
```

15. MQTT 协议（物联网可用）

协议参考文档为：《mqtt-v3.1.1-os.pdf》

使用 PUBLISH 来发送消息，参数映射如下：

Mqtt 字段	参数	解释
TopicName	手机号	
Payload	短信内容	Utf8 编码

注意限制为：Remaining Length 最大值为 16383。

可以通过订阅"receive"，来接收上行和状态报告
在接收时，TopicName 有两种格式：
“M13612345678”表示一个上行信息，payload 为消息内容，utf8 编码
“R13612345678”表示一个状态报告，payload 为状态描述，utf8 编码
KeepAlive 参数忽略。
以下为 golang 演示代码，很多地方做了简化处理，请根据实际情况细化：

```
var clientIdentifier = "dev001"
var username = "test"
var password = "123456"

var mobile = "13612345678"
var message = "短信测试【测试】"

// 建立链接
conn, _ := net.Dial("tcp", "192.168.1.100:7891")

// CONNECT 报文
var clientLen = len(clientIdentifier)
var userLen = len(username)
var passLen = len(password)
var remainingLength = 10 + clientLen + 2 + userLen + 2 + passLen + 2
var buf = make([]byte, remainingLength + 2)
buf[0] = 1
buf[1] = byte(remainingLength)
buf[2] = 0
buf[3] = 4
copy(buf[4:], "MQTT")
buf[8] = 4
buf[9] = 0xC2
buf[10] = 0
buf[11] = 30
buf[12] = 0
buf[13] = byte(clientLen)
copy(buf[14:], clientIdentifier)
var off = 14 + clientLen
buf[off] = 0
buf[off+1] = byte(userLen)
copy(buf[off+2:], username)
off = off + 2 + userLen
buf[off] = 0
buf[off+1] = byte(passLen)
copy(buf[off+2:], password)
off = off + 2 + passLen
```

```
// 发送 CONNECT 请求
conn.Write( buf )

// 读取 CONNACK
var ack = make([]byte, 4)
conn.Read(ack)

// 准备 PUBLISH
var msg = []byte(message)
var msgLen =    len(msg)
var mobileLen = len(mobile)
var packetIdentifier = 1
remainingLength = 2 + mobileLen + 2 + msgLen
buf = make([]byte, remainingLength + 2)
buf[0] = 0x32
buf[1] = byte(remainingLength)
buf[2] = 0
buf[3] = byte(mobileLen)
copy(buf[4:], mobile)
off = 4 + mobileLen
buf[off] = byte(packetIdentifier >> 8)
buf[off+1] = byte(packetIdentifier)
copy(buf[off+2:], msg)

// 发送短信
conn.Write( buf )

// 读取应答
ack = make([]byte, 4)
conn.Read(ack)

// 发送 DISCONNECT
buf = make([]byte, 2)
buf[0] = 0xE0
conn.Write( buf )

conn.Close()
```

16. 错误码定义

以下为系统定义的错误码列表

错误码	说明
9002	未知命令
9012	短信消息内容错误
9013	目标地址错误
9014	短信内容太长
9015	路由错误
9016	没有下发网关
9017	定时时间错误
9018	有效时间错误
9019	无法拆分或者拆分错误
9020	号码段错误
9021	消息编号错误，这个和 PacketIndex 参数有关
9022	用户不能发长短信(EsmClass 错误)
9023	ProtocolID 错误
9024	结构错误，一般是指长短信
9025	短信编码错误
9026	内容不是长短信
9027	签名不对
9028	目标网关不支持长短信
9029	路由拦截
9030	目标地址(手机号)太多
9031	目标地址(手机号)太少
9032	发送速度太快
9101	验证失败，一般和用户名/密码/IP 地址相关
9102	没有填写用户名
9103	名字没找到
9104	IP 地址不对
9105	超过最大连接数，就是 tcp 连接数，http 也是一样的
9106	协议版本错误
9107	帐号无效，比如过期/禁用
9902	网关无此能力
9903	二进制数据太长了；如网关没有特别说明，一般不能超过 140，
9904	网关不支持 EsmClass 字段，或等同字段
9905	网关不支持 ProtocolID 字段，或等同字段
9906	网关不支持 UDHI 字段，或等同字段
9907	网关支持 Letter 字段发送，但短信记录没有 letter
9908	网关不存在
9909	网关没有应答
9910	网关不支持该短信编码

9911	区域错误
9401	计费错误
9402	非法内容
9403	黑名单
9404	
9405	Api 帐号丢失
9406	配置拒绝，就是帐号设置了拒绝标记
9407	帐号没有生成时间,这个属于非法帐号
9408	消息超时，超过短信或帐号或系统设置的生存时间
9409	由约束规则拒绝
9410	状态报告超时
9411	
9412	帐号无效
9413	重发拦截
9414	转发时丢弃，比如该通道已经废弃
9415	人工审核失败
9416	可能是诈骗信息
9417	不匹配模板
9418	拒绝审核（审核功能可能关闭）
9419	超过该手机号码的日发送次数限制
9501	非法目标地址，即手机号
9502	消息无法投入队列
9601	上行路由失败
9602	超过最大重试
9701	通知失败
9702	处理配置错误
9801	投递地址错
9802	无法连接到服务器
9803	投递发送数据失败
9804	投递接收结果失败

仅供参阅

17. 名词解释

12.1. data_coding

消息编码,smpp 之 data_coding，8bits。一般 0 表示英文,8 表示 unicode,15 表示中文。如果要发送繁体、日文、韩文等文字，请使用 8，并将内容转为 UCS2(BigEndian)格式；即每个字符对应 2 个字节，高字节在前。

12.2. esm_class

smpp 之 esm_class, 8bits；一般不用

12.3. content

默认 HEX 编码之消息内容；例如内容’ABC’经 HEX 后为’414243’；也可以指定编码格式。详细请参考 transfer_encoding 的说明。

12.4. protocol_id

smpp 之 protocol_id,cmpp 之 tp_pid，8bits；一般不用

12.5. dest_addr

在发送时为手机号码；多个号码用分号(半角)分割。请不要超过 100 个。
在接收上行信息时为扩展码，一般为数字字符串。
如果是不同手机号不同内容群发，则该参数为空，或者客户自定义分割符。可参考 http 接口说明。

12.6. source_addr

在发送时为扩展码，必须以账号设定的开头；可以不填写。
在接收上行信息时为手机号码

12.7. link_id

目前不用

12.8. external_id

外部编码,长整型；客户可以自行填写，状态报告时返回

12.9. registered_delivery

是否需要状态报告；0 表示不需要；1 表示需要

12.10. username

用户名

12.11. password

密码。**请不要在通讯中直接使用密码，如必须使用，请务必提供 IP 地址进行锁定。在 http 接口中，直接或间接的形式都建议作 URLEncode 处理。**

12.12. schedule_time

定时发送时间；格式 yyyyMMddHHmmss；可不填

12.13. module

模块名，一般不用填写

12.14. priority

优先级, 取值范围 0~1,一般不设；账号优先级由系统控制，并可以根据规则自行调整。

12.15. validity_period

有效期；格式 yyyyMMddHHmmss；可不填

12.16. response_format

控制返回格式：
0：默认格式，即文本格式。
1：xml，xml 格式中的元素名同 HTTP 接口命名。

2：json 格式

12.17. result

错误码，0 表示成功,其他错误代码；在 0 时该字段可省略

12.18. message_id

消息的唯一编号（如果是群发则相当于批号），发送时返回；用于在状态报告接收时进行匹配。

12.19. operation

取值 mo 或者 dr 表明这是一个上行短信或者状态报告

12.20. status

状态报告描述字符串

12.21. submit_date

消息提交时间

12.22. done_date

消息完成时间

12.23. receipt

状态报告错误码，0 表示用户接收成功。

12.24. block_index

消息发生拆分，该值表明这是第几段消息

12.25. dest_index

在群发时该字段表示，第几个号码。如果群法时有些号码是非法的，则这些号码没有编号。也就是说只给那些服务器接收并产生下发记录的消息编号。

12.26. timestamp

如果该字段存在，服务器将认为 password 由如下方式计算得出：
password=Base64(MD5(username+password+timestamp+content))
其中 username，password，timestamp 以 utf-8 处理成字节数组。拼接时是按字节数组拼接(content 是按原始短信字节数组处理，并非编码后的字符串)；MD5 之后得到 16 字节(非长度为 32 的字符串)，再使用 Base64 编码。如果 content 不存在，则不参与计算。
该字段格式 yyyyMMddHHmmss
java 的示例代码如下：

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
String timestamp = sdf.format(new Date() );
MessageDigest md5 = MessageDigest.getInstance("MD5");
md5.update( username.getBytes("utf8") );
md5.update( password.getBytes("utf8") );
md5.update( timestamp.getBytes("utf8") );
if (message != null)
    md5.update( message.getBytes("utf8") );
password = Base64.encodeBase64String( md5.digest() );
```

12.27. fetch_size

可以一次提取的数量，这个数量不能超过服务器段的配置。如果超过或者无效，将以服务器端的配置为准。

12.28. transfer_encoding

短信内容的传输编码
0：HEX 编码格式(这是默认编码)
1：Base64 编码格式
2：URLEncode 编码（即针对字节进行 URLEncode 编码）
3：URLEncode+UTF8（即原始文本用 UTF8 转字节后进行 URLEncode 编码）
举例如下：

	tf 取值	英文消息：“abc”	中文消息：“您好”
HEX	0 或者不设置	616263	c4fabac3

Base64	1	YWJj	xPq6ww==
URLEncode	2	abc	%C4%FA%BA%C3
URLEncode+UTF8	3	abc	%E6%82%A8%E5%A5%BD

说明：在以上例子中，英文消息请设置 dc=0，中文消息设置 dc=15（gbk）。
如果发送日文，韩文等其他特殊语言字符，请设置 dc=8，并使用 tf=3 的方式。

12.29. *

18. 安全性

对于银行金融业可能需要较高的安全级别；以上接口协议均支持 SSL/TLS（基于 HTTP 协议的即为 HTTPS，推荐使用 SSL3.0 以上）。

19. 客户端业务支持

以下几个字段可以用于客户端的业务关联处理:

字段	类型	说明
externalId	integer(8)	一般可以用于对应唯一一条短信，也可以用于对应业务，比如某次群发。通过仔细设计，比如分成两个 integer(4)，一个用来表示用户编号，一个用来表示短信编号。
module	string(32)	可以用来表示业务类型。或则表示某个部门发送的短信，经过特殊设计还可以携带更多的业务信息。

20. 备注

本系统提供统一的对外服务地址和端口，并支持文档所述所有协议。
标准协议均可使用各运营商提供的协议开发包进行对接。