

# HTTP 协议短信接口文档

版本：V2.0.0

最新修改时间：2016-04-20

# 目 录

|     |                            |    |
|-----|----------------------------|----|
| 1   | 账户短信发送.....                | 4  |
| 1.1 | 下发接口地址.....                | 4  |
| 1.2 | 短信下发参数.....                | 4  |
| 1.3 | 下发返回参数说明.....              | 5  |
| 2   | 个性化短信群发.....               | 5  |
| 2.1 | 个性化短信群发接口地址.....           | 5  |
| 2.2 | 个性化短信发送参数.....             | 5  |
| 2.3 | 发送返回说明.....                | 6  |
| 2.4 | 提交方式说明.....                | 7  |
| 3   | 账户余额查询.....                | 7  |
| 3.1 | 余额接口地址.....                | 7  |
| 3.2 | 余额接口参数.....                | 7  |
| 3.3 | 余额返回参数说明.....              | 7  |
| 4   | 状态报告推送.....                | 7  |
| 4.1 | 状态报告推送方式.....              | 8  |
| 4.2 | 状态报告推送样例.....              | 8  |
| 4.3 | 状态报告响应结果.....              | 8  |
| 5   | 客户主动获取状态报告设计.....          | 9  |
| 5.1 | 访问接口地址.....                | 9  |
| 5.2 | 访问参数说明.....                | 9  |
| 5.3 | 返回参数说明.....                | 9  |
| 5.4 | 状态报告推送样例.....              | 9  |
| 6   | 上行信息推送.....                | 10 |
| 6.1 | 上行信息推送方式.....              | 10 |
| 6.2 | 上行信息推送样例.....              | 11 |
| 6.3 | 上行信息响应结果.....              | 11 |
| 7   | 客户主动获取上行信息设计.....          | 11 |
| 7.1 | 访问接口地址.....                | 11 |
| 7.2 | 访问参数说明.....                | 11 |
| 7.3 | 返回参数说明.....                | 12 |
| 7.4 | 上行信息推送样例.....              | 12 |
| 8   | 附录（java 下发短信 demo）.....    | 13 |
| 9   | 附录（java 个性化短信群发 demo）..... | 13 |
| 10  | 附录（java 查询余额 demo）.....    | 15 |

|                                |    |
|--------------------------------|----|
| 11 附录 (java 接收上行 demo).....    | 16 |
| 12 附录 (主动获取上行/状态报告 demo) ..... | 17 |
| 13 附录 (关于域名解析的相关说明) .....      | 17 |

## 1 账户短信发送

为了降低客户技术开发难度，根据客户需求，此文档提供 HTTP 协议接口。客户可以使用 POST 方式调用该接口进行短信下发。

文档中访问域名地址由客服经理提供。

### 1.1 下发接口地址

1. 服务器接口地址：[http://域名:8080/sms\\_send2.do](http://域名:8080/sms_send2.do)
2. POST 方式调用，请您参考此文档最后附录（java 下发短信 demo）。

### 1.2 短信下发参数

参数说明：

| 参数名称         | 参数说明   | 备注   |
|--------------|--|--|
| corp_id      | 访问接口账户 id  | 由系统管理员开通   |
| corp_pwd     | 访问接口账户密码   |  |
| corp_service | 业务代码   |  |
| mobile       | 下发目的手机号码   | 多个号码之间用英文逗号隔开，且一次群发总号码数不能超过 200 个                |
| msg_content  | 短信内容   | 短信内容长度不超过 1000 个汉字（包括 1000 字），每个英文或阿拉伯字符也算 1 个汉字 |
| corp_msg_id  | 用户发送短信时自己定义的短信 id，用于区分状态报告。 <b>参数名必须填写，参数值可为空。</b>   | 总长度不能超过 50 个字符                                   |
| ext          | 用户自行分配扩展号。 <b>参数名必须填写，参数值可为空。</b><br>该参数是显示在接收手机上的主叫尾号，可用于上行信息匹配，例：<br>我方给合作方开通特服号为：10657532521924，合作方在发送信息时可随意填写扩展号（需为 0-9 数字）ext=8888，那么用户收到信息时显示的下发号码就是 106575325219248888。 | 通道本身特服号加上用户自己分配扩展号的总长度不能超过 20 位。                 |

**注意：**

- 1) 发送速度：接口访问没有时间间隔限制。建议用户等到接口返回值以后再进行下一次调用。
- 2) IP 访问：如果用户开账户时指定 IP，则此接口只接收指定 IP 的发送请求。IP 支持单 IP，多 IP，IP 号段及无 IP。
- 3) 内容长度：本接口支持每条短信内容的最大长度不超过 1000 个汉字（包括 1000 字）。但是具体长度需要视具体通道而定。
- 4) 编码格式：提交信息时，汉字应转成 GB2312 提交，否则手机用户收到会显示乱码。

**1.3 下发返回参数说明**

| 返回代码 | 代码说明                                     |
|------|--|
| 0#数字 | 提交成功。数字：提交成功的手机数量                        |
| 100  | 余额不足                                     |
| 101  | 账号关闭                                     |
| 102  | 短信内容超过 1000 字（包括 1000 字）或为空              |
| 103  | 手机号码超过 200 个或合法手机号码为空或者与通道类型不匹配          |
| 104  | corp_msg_id 超过 50 个字符或没有传 corp_msg_id 字段 |
| 106  | 用户名不存在                                   |
| 107  | 密码错误                                     |
| 108  | 指定访问 ip 错误                               |
| 109  | 业务代码不存在或者通道关闭                            |
| 110  | 扩展号不合法                                   |
| 9    | 访问地址不存在                                  |

**注意：**

- 1) 长短信：对于长短信，“#”后面的数字代表用户提交的条数，而不是实际扣费的条数。例如用户发送了 190 字的短信三条，返回的是 0#3, 后台根据拆分后的结果，计费为 9 条。
- 2) 0#数字：此返回值数字代表成功提交到我方平台的数据。失败条数不会显示，失败号码也不会显示。

**2 个性化短信群发****2.1 个性化短信群发接口地址**

接口地址为：<http://域名:8080/hSmsSend.do>

**2.2 个性化短信发送参数**

| 参数名称         | 参数定义     | 参数说明       | 备注   |
|--------------|----------|------------|--|
| corp_id      | 企业 id    | 企业身份标示     |  |
| corp_pwd     | 企业的密码    | 对应 id 标示   |  |
| corp_service | 业务代码     | 用来发送短信的业务  |  |
| total_count  | 总条数      | 本次发送总条数    | 用来与接口拼凑完成后的实际条数进行校验。 <b>不大于 200</b>  |
| send_param   | 发送的短信的参数 | 发送的实际内容    | 1、每条个性化短信内容不能超过 1000（包括 1000 字）字并且不能为空。<br>2、主叫扩展号码只支持数字。主叫扩展号码+主叫号码<20 位。 |
| ctrl_param   | 控制参数     | 控制个性化提交的模式 | 暂时无用   |

**说明：**

- 1) send\_param格式：手机号码&split&短消息序号&split&主叫扩展号码&split&短消息内容。
- 2) 每条短信之间&group&以分隔。以&group&拆分后的短信条数必须等于total\_count，如有差别，则全部失败。
- 3) 主叫扩展号码不限制长度，但是需保证通道接入号+主叫扩展号≤20，总长度超过20位则发送失败
- 4) 样例：

send\_param=13601025412&split&20110608122535001&split&256&split&尊敬的会员张三您好，您的账户余额为 20 元  
&group&13821056854&split&2011060812255425003&split&257&split&尊敬的会员张三您好，您的账户余额为 30.2 元

**2.3 发送返回说明**

| 参数   | 说明             | 备注   |
|------|----------------|--|
| 0#数字 | 提交成功#提交成功的短信数量 | 1、通道支持长短信整条提交则返回长短信数；若不支持整条提交则返回拆分后短信总数；<br>2、0#0：表示此次个性化短信全部提交失败。 |
| 100  | 余额不足           |  |
| 101  | 账号关闭           |  |
| 106  | 用户名不存在         |  |
| 107  | 密码错误           |  |

|     |   |                              |
|-----|---|------------------------------|
| 108 | 指定访问的 IP 错误   | 接口支持绑定单 IP, 多 IP, IP 号段及无 IP |
| 109 | 业务不存在   |                              |
| 110 | 扩展不合法   |                              |
| 114 | 接口提交应为 POST, 不支持 GET  |                              |
| 115 | total_count 与实际短信条数无法匹配, 即, 实际短信条数与 total_count 不一致。如果要返回此参数, 则本次提交的所有短信作废, 不入库 |                              |
| 116 | 个性化短信提交个数超过 200 条   |                              |
| 9   | 访问地址不存在   |                              |

## 2.4 提交方式说明

考虑到短信内容长度拼接后较长, 接口支持 HTTP 协议中的 POST 方法, 不支持 GET 方法。并且由于个性化参数中本身带有&符号, 所以不能使用 curl 命令测试。具体请您参考此文档最后附录 (java 发送个性化短信 demo) 进行测试。

## 3 账户余额查询

### 3.1 余额接口地址

余额接口地址: [http://域名:8080/sms\\_count2.do](http://域名:8080/sms_count2.do)

POST 方式调用, 请您参考此文档附录 10 (java 查询余额 demo)。

### 3.2 余额接口参数

| 参数名称     | 参数说明      | 备注       |
|----------|-----------|----------|
| corp_id  | 访问接口账户 id | 由系统管理员开通 |
| corp_pwd | 访问接口账户密码  | 由系统管理员设定 |

**注意事项:** 接口访问频率限制为 5 次/秒, 即两次访问间隔需在 200ms 以上

### 3.3 余额返回参数说明

1) 正常返回状态: ok#余额

2) 错误返回状态:

| 参数  | 说明             | 备注                           |
|-----|----------------|------------------------------|
| 101 | 账号关闭           |                              |
| 104 | 两次访问间隔小于 200ms |                              |
| 106 | 用户名不存在         |                              |
| 107 | 密码错误           |                              |
| 108 | 指定访问的 IP 错误    | 接口支持绑定单 IP, 多 IP, IP 号段及无 IP |
| 9   | 访问地址不存在        |                              |

## 4 状态报告推送

## 4.1 状态报告推送方式

状态报告内容：用以确认短信是否下发成功。我方主动将状态报告内容以 GBK 编码形式推送给用户。

状态报告推送地址：由客户提供状态报告接收地址，类似 `http://.....dfdf.do` 或者是 `http://.....dfdf.aspx` 等等。

## 4.2 状态报告推送样例

推送的数据样例(为一整条 xml 格式的字符串，注：是一整条字符串，不是 xml 文档)

```
<?xml version="1.0" encoding="GBK" ?>
```

```
<reports>
  <report>
    <corp_id>test</corp_id>
    <mobile>13810000001</mobile>
    <sub_seq>0</sub_seq>(0 表示整条，其余数字表示一条长短信被拆分的第
    几条)
    <msg_id>12345asd</msg_id>
    <err>2</err>(表示具体的错误码，其中 err 为 0 或 000 均表示为成功)
    <fail_desc>undeliver</fail_desc>(表示具体的错误描述)
    <report_time>2010-07-02 00:00:00</report_time>
  </report>
  <report>
    <corp_id>test</corp_id>
    <mobile>13810000002</mobile>
    <sub_seq>0</sub_seq>
    <msg_id>12345asd</msg_id>
    <err>2</err>(表示具体的错误码，其中 err 为 0 或 000 均表示为成功)
    <fail_desc>undeliver</fail_desc>
    <report_time>2010-07-02 00:00:00</report_time>
  </report>
</reports>
```

### 备注：

一个 reports 最多可包含 100 个 report 节点，即一个状态报告 xml 字符串最多可包含 100 个状态报告。

状态报告可根据 mobile 和 msg\_id 唯一标识。

## 4.3 状态报告响应结果

用户接收到我方推送的状态报告时，需返回响应结果。

0: 接收成功。

## 4.4 状态推送策略

推送策略：

我方设置响应超时时间为 20 秒。如果我方在 20 秒之内未收到、或未收到正确的用户响应结果，则按用户接收失败处理。处理方式如下：我方重复推送该条状态报告，最多连续推送三次，每次间隔 2s。三次推送失败后，此条状态报告信息不再继续推送。

#### 重复推送机制：

如因状态报告推送地址配置有误、或网络等外部原因导致状态报告推送失败的，如需要，可联系我方技术人员再次推送，推送策略同上。

#### 建议：

建议当接收到状态报告 xml 字符串时，先给我方回复响应，同时将信息保存起来，之后再进行处理工作。否则，边接收边处理，处理后再给响应，会影响状态报告推送效率，造成积压。

## 5 客户主动获取状态报告设计

### 5.1 访问接口地址

访问地址：[http://域名:8080/post\\_report.do](http://域名:8080/post_report.do)

访问方式：post 方式，具体请见此文档最后附录：主动获取上行/状态报告 demo。

编码：GBK 编码形式。

### 5.2 访问参数说明

| 参数名称     | 参数说明    | 备注       |
|----------|---------|----------|
| corp_id  | 一级账户用户名 | 由系统管理员设定 |
| corp_pwd | 一级账户密码  | 由系统管理员设定 |

### 5.3 返回参数说明

| 返回代码      | 代码说明                            |
|-----------|---------------------------------|
| 0         | 暂时没有待推送的数据                      |
| 9         | 访问地址不存在                         |
| -11       | 账户关闭                            |
| -16       | 用户名错误或用户名不存在                    |
| -17       | 密码错误                            |
| -18       | 不支持客户主动获取                       |
| -19       | 用户访问超过我方限制频率（间隔 200 毫秒访问一次）     |
| 108       | 指定访问 IP 错误                      |
| Xml 格式字符串 | 有待推送的状态报告，并且状态报告以 xml 格式的字符串返回。 |

### 5.4 状态报告获取样例

用于客户查询每条短信发送的状态是否成功。

推送的数据样例(为一整条 xml 格式的字符串，注：是一整条字符串，不是 xml 文档)

```
<?xml version="1.0" encoding="GBK" ?>
```

```

<reports>
  <report>
    <corp_id>test</corp_id>
    <mobile>13810000001</mobile>
    <sub_seq>0</sub_seq>(0 表示整条，其余数字表示一条长短信被拆分的第
几条)
    <msg_id>12345asd</msg_id>
    <err>2</err> (表示具体的错误码，其中 err 为 0 或 000 均表示为成功)
    <fail_desc>undeliver</fail_desc> (表示具体的错误描述)
    <report_time>2010-07-02 00:00:00</report_time>
  </report>
  <report>
    <corp_id>test</corp_id>
    <mobile>13810000002</mobile>
    <sub_seq>0</sub_seq>
    <msg_id>12345asd</msg_id>
    <err>2</err> (表示具体的错误码，其中 err 为 0 或 000 均表示为成功)
    <fail_desc>undeliver</fail_desc>
    <report_time>2010-07-02 00:00:00</report_time>
  </report>
</reports>

```

**备注:**

一个 reports 最多可包含 100 个 report 节点,即一个状态报告 xml 字符串最多可包含 100 个状态报告。

## 5.5 状态报告获取策略

**获取策略:**

客户可访问我方提供的接口地址，主动获取状态报告信息，一个状态报告 xml 字符串最多可包含 100 个状态报告，每次获取间隔时间要大于等于 200ms，待获取的信息在我方平台最多保留三天，超过三天则自动转移为历史数据，最多保留三个月。

**重复获取:**

如因特殊原因未取成功的，如需要，可联系我方技术人员进行处理，可重新获取，获取策略同上。

## 6 上行信息推送

### 6.1 上行信息推送方式

**上行信息内容:** 手机用户给发送方回复的信息。我方主动将上行信息内容以 GBK 编码形式推送给用户。

**上行接收地址:** 由客户提供接收上行信息的 url 地址,类似 http://.....dfdf.do 或者是 http://.....dfdf.aspx 等等。

## 6.2 上行信息推送样例

推送的数据样例：(为一整条 xml 格式的字符串，注：是一整条字符串，不是 xml 文档)

```
<?xml version="1.0" encoding="GBK" ?>
<delivers>
  <deliver>
    <corp_id>test</corp_id>
    <mobile>13810000000</mobile>
    <ext>8888</ext> (对应下发时的 ext 参数，根据手机用户回复短信（即上行信息）
    的 ext 的值， 匹配客户或者客户下发的信息)
    <time>2010-07-02 00:00:00</time>
    <content>收到</content>
  </deliver>
</delivers>
```

### 注意：

一条上行 xml 字符串最多可包含 10 个 < deliver ></ deliver> 节点。  
上行信息根据 mobile 和完整接入号唯一标识一个账户。

## 6.3 上行信息响应结果

用户接收到我方推送的上行信息时，需返回响应结果。

0: 接收成功。

### 注意：

## 6.4 上行推送策略

### 推送策略：

我方设置响应超时时间为 20 秒。如果我方在 20 秒之内未收到、或未收到正确的用户响应结果，则按用户接收失败处理。处理方式如下：我方重复推送该条上行信息，最多连续推送三次，每次间隔 2s。三次推送失败后，此条上行信息不再继续推送。

### 重复推送机制：

如因上行推送地址配置有误、或网络等外部原因导致上行信息推送失败的，如需要，可联系我方技术人员再次推送，推送策略同上。

### 建议：

建议当接收到上行 xml 字符串时，先给我方回复响应，同时将信息保存起来，之后再进行处理工作。否则，边接收边处理，处理后再给响应，会影响上行推送效率，造成积压。

# 7 客户主动获取上行信息

## 7.1 访问接口地址

访问地址：[http://域名:8080/post\\_deliverMsg.do](http://域名:8080/post_deliverMsg.do)

访问方式：post 方式，具体请见此文档最后附录：主动获取上行/状态报告 demo。

编码：GBK 编码形式。

## 7.2 访问参数说明

| 参数名称     | 参数说明    | 备注       |
|----------|---------|----------|
| corp_id  | 一级账户用户名 | 由系统管理员设定 |
| corp_pwd | 一级账户密码  | 由系统管理员设定 |

### 7.3 返回参数说明

| 返回代码      | 代码说明  |
|-----------|---|
| 0         | 暂时没有待推送的数据  |
| 9         | 访问地址不存在   |
| -11       | 账户关闭  |
| -16       | 用户名错误或用户名不存在  |
| -17       | 密码错误  |
| -18       | 不支持客户主动获取   |
| -19       | 用户访问超过我方限制频率（间隔 200 毫秒访问一次）   |
| 108       | 指定访问 IP 错误  |
| Xml 格式字符串 | 有待推送的上行信息，并且上行信息以 xml 格式的字符串返回。一条上行 xml 字符串可包含 10 个< deliver ></ deliver>节点 |

### 7.4 上行信息获取样例

推送的数据样例：(为一整条 xml 格式的字符串，注：是一整条字符串，不是 xml 文档)

```
<?xml version="1.0" encoding="GBK" ?>
<delivers>
  <deliver>
    <corp_id>test</corp_id>
    <mobile>13810000000</mobile>
    <ext>8888</ext> (对应下发时的 ext 参数，根据手机用户回复短信（即上行信息）
    的 ext 的值， 匹配客户或者客户下发的信息)
    <time>2010-07-02 00:00:00</time>\
    <content>收到</content>
  </deliver>
</delivers>
```

#### 注意：

一条上行 xml 字符串最多包含 10 个< deliver ></ deliver>节点。

### 7.5 上行获取策略

#### 获取策略：

客户可访问我方提供的接口地址，主动获取上行信息，一个上行 xml 字符串最多可包含 10 个节点，每次获取间隔时间要大于等于 200ms，待获取的信息在我方平台最多保留三天，超过三天则自动转移为历史数据，最多保留三个月。

#### 重复获取：

如因特殊原因未取成功的，如需要，可联系我方技术人员再次进行处理，可重新获取，获取策略同上。

## 8 附录（JAVA 下发短信 DEMO）

```

public static void main(String[] args) throws Exception {
    HttpClient client = new HttpClient();
    PostMethod post = new PostMethod("http://域名:8080/sms_send2.do");
    post.setRequestHeader("Content-Type","application/x-www-form-urlencoded;charset=gbk");//在头文件中设置转码
    NameValuePair[] data ={
        new NameValuePair("corp_id","访问接口账户id"),
        new NameValuePair("corp_pwd","访问接口账户密码"),
        new NameValuePair("corp_service","业务代码"),
        new NameValuePair("mobile","下发目的手机号码"),
        new NameValuePair("msg_content","下发短信内容"),
        new NameValuePair("corp_msg_id","短信Id"),
        new NameValuePair("ext","扩展小号")
    };
    post.setRequestBody(data);
    client.executeMethod(post);
    Header[] headers = post.getResponseHeaders();
    int statusCode = post.getStatusCode();
    System.out.println("statusCode:"+statusCode);
    for(Header h : headers){
        System.out.println(h.toString());
    }
    String result = new String(post.getResponseBodyAsString());
    System.out.println(result);
    post.releaseConnection();
}

```

## 9 附录（JAVA 个性化短信群发 DEMO）

```

public static int send_cat(String corp_id, String corp_pwd, String url,
    String corp_service, int total_count, String send_param,int ctrl_para){
    int result =-1;
    HttpClient client = new HttpClient();
    PostMethod method = new PostMethod(url);
    method.setRequestHeader("Content-Type","application/x-www-form-urlencoded;charset=gbk");
    try{
        method.addParameter("corp_id", corp_id);
        method.addParameter("corp_pwd", corp_pwd);
        method.addParameter("corp_service", corp_service);
        method.addParameter("total_count",total_count+"");
    }
}

```

```
        method.addParameter("send_param",send_param);
        method.addParameter("ctrl_param", ctrl_param+""");
        client.executeMethod(method);
        String sendResult = method.getResponseBodyAsString();
        System.out.println("====="+sendResult);
    }catch (Exception e) {
        e.printStackTrace();
        result = -1;
    }finally{
        try{
            method.releaseConnection();
        }catch (Exception e) {
            e.printStackTrace();
        }
    }
}
return result;
}

public static void main(String[] args) {
    long start=System.currentTimeMillis();
    System.out.println(start);
    String tel="手机号码";
    String content=短信内容";
    String msg_id="短消息序号";
    String code="主叫扩展号";
    String split("&split&";
    String group("&group&";
    String send_param = "";
    String[] array={tel,msg_id,code,content};
    StringBuffer sb=null;
    int length=2;
    for (int i = 0; i < length; i++) {
        sb=new StringBuffer();
        for (int j = 0; j < array.length; j++) {
            if((j+1)==array.length){
                sb.append(array[j]+i);
            }else{
                sb.append(array[j]+split);
            }
        }
    }
}
```

```

    }
    if((i+1)==length){
        send_param+=sb.toString();
    }else {
        send_param+=sb.toString()+group;
    }
    sb=null;
}
System.out.println(System.currentTimeMillis()-start);
System.out.println(send_param);

char[] c = send_param.toString().toCharArray();
System.out.println("charlength= " + c.length);

String corp_id = "企业 id";
String corp_pwd = "企业密码";
String url = "http://域名:8080/hSmsSend.do";
String corp_service="业务代码";
int total_count=length;
AntoSmsTest.send_cat(corp_id, corp_pwd, url,corp_service,total_count,
send_param,1);
}

```

## 10 附录（JAVA 查询余额 DEMO）

```

public static void main(String[] args)throws Exception{
    HttpClient client = new HttpClient();
    PostMethod post = new PostMethod("http://域名:8080/sms_count2.do");
    post.setRequestHeader("Content-Type","application/x-www-form-urlencoded;charset=gbk");//在头文件中设置转码
    NameValuePair[] data ={
        new NameValuePair("corp_id","访问接口账户id "),
        new NameValuePair("user_id","访问接口账户id "),
        new NameValuePair("pwd", "访问接口账户密码")
    };
    post.setRequestBody(data);
    client.executeMethod(post);
    Header[] headers = post.getResponseHeaders();
    int statusCode = post.getStatusCode();
    System.out.println("statusCode:"+statusCode);
    for(Header h : headers){
        System.out.println(h.toString());
    }
}

```

```

    }
    String result = new String(post.getResponseBodyAsString());
    System.out.println(result);
    post.releaseConnection();
}

```

## 11 附录 (JAVA 接收上行 DEMO)

介绍：我方推送过去的 xml 格式的上行字符串，参照如下

```

String str="<?xml version='1.0' encoding='GBK'?>"
<delivers>" + "<deliver><corp_id>test</corp_id><mobile>13860497631</mobile>
    <ext>2010</ext><time>2010-07-02 00:00:00</time><content>您好 tuisong 测试
    222</content></deliver>" + "</delivers>";

```

1.通过访问合作方给提供 url 地址，调用到合作方的 URL，然后我方推送上行信息的 xml 字符串过去，合作方接收。

2.此条 xml 字符串，是存在 request 的内存里。

3.需要接收方先取得 request 对象(注意这里 request 对象无参数，是直接存在 request 内存里，需要先从 request 里获取输入流，然后进行读流操作，读出推送内容)

参考如下：

```

protected void doPost(HttpServletRequest request, HttpServletResponse resp)
throws ServletException, IOException {
    InputStream in = request.getInputStream();
    BufferedInputStream buf = new BufferedInputStream(in);
    byte[] buffer = new byte[1024];
    StringBuffer data =new StringBuffer();
    int a ;
    while((a = buf.read(buffer))!= -1){
        data.append(new String(buffer,0,a,"gbk"));
    }
    String getData = data.toString();
    System.out.println(getData ); //显示接收的字符串
    if(getData.equals(null)||getData.equals("")){
        resp.getWriter().write("9"); //如果接收失败，返回 9
    }
    else{
        resp.getWriter().write("0"); //如果接收成功，返回 0
    }
}

```

```

protected void doGet(HttpServletRequest request, HttpServletResponse resp)
throws ServletException, IOException {
    (request,resp);
}

```

4. 解析取得的 xml 格式的字符串，取得推送上行内容的各个节点的值，DOM4j 解析参考如下：

```

        Document doc = DocumentHelper.parseText(字符串);
        Element root = doc.getRootElement();
        List<Element> deliverElements = root.elements("deliver");
    for(Element deliver:deliverElements){
        String corp_id = deliver.elementText("corp_id");
        String mobile = deliver.elementText("mobile");
        String ext = deliver.elementText("ext");
        String time = deliver.elementText("time");
        String content = deliver.elementText("content");
        System.out.println(corp_id+":"+mobile+":"+ext+":"+time+":"+content);
    }

```

## 12 附录（主动获取上行/状态报告 DEMO）

```

HttpClient client = new HttpClient();
PostMethod post = new PostMethod("主动获取的 URL 地址");
post.setRequestHeader("Content-Type","application/x-www-form-urlencoded;charset=gbk");//

```

在头文件中设置转码

```

NameValuePair[] data = {
    new NameValuePair("corp_id", "访问接口账户 id"),//一级账户 id
    new NameValuePair("corp_pwd", "wehpns"),//一级账户密码密
};
post.setRequestBody(data);
client.executeMethod(post);
Header[] headers = post.getResponseHeaders();
int statusCode = post.getStatusCode();
System.out.println("statusCode:"+statusCode);
for(Header h : headers){
    System.out.println(h.toString());
}
String result = new String(post.getResponseBodyAsString());
System.out.println(result);
post.releaseConnection();

```

## 13 附录（关于域名解析的相关说明）

1. **请勿**直接在您的服务器系统 `hosts` 文件上写域名和 IP 对应关系。

如：www.example.com 93.184.216.34

以免我司平台域名解析切换成新 IP 后，您的服务器仍然解析成旧的 IP 地址。

2. **请勿**在您的服务器系统上配置公司内部 DNS 服务器，建议配置成知名的 DNS 地址。

```
如(114.114.114.114)
# cat /etc/resolv.conf
nameserver 114.114.114.114
nameserver 119.29.29.29
```

内部 DNS 服务器或当地运营商的 DNS 服务器大多 DNS 解析变更时会有更新延迟(一般公司内部 DNS 的解析延迟时间可能多达一天), 以免我司平台域名解析切换成新 IP 后, 您的服务器在很长一段时间内仍然解析成旧的 IP 地址, 造成无法访问我司短信平台。我司使用 Dnspod 域名解析企业服务, 一般更改解析后, 知名的 DNS 解析服务器会在 2 分钟后更新。

### 3. 如何确认在您的服务器上解析的域名是最新的解析 IP:

访问 <http://ping.chinaz.com/> , 输入我司短信平台域名, 查看全国各地解析的 IP 。  
如:



The screenshot shows a web-based ping test interface. At the top, there is a text input field labeled '请输入IP或域名:' containing the URL 'http://www.example.com/'. To the right of the input field is a blue button labeled '查询'. Below the input field, there is a list of regions with their corresponding IP addresses. The IP address '93.184.216.34' is highlighted in red in the '香港[电信]' row.

| 地区       | IP地址          |
|----------|---------------|
| 贵州[电信]   | 93.184.216.34 |
| 福建福州[电信] | 93.184.216.34 |
| 香港[电信]   | 93.184.216.34 |
| 江苏[电信]   | 93.184.216.34 |
| 云南昆明[电信] | 93.184.216.34 |

在您的服务器上 ping 我司平台域名, 查看解析 IP:

```
[root@nn1 ~]# ping www.example.com
PING www.example.com (93.184.216.34) 56(84) bytes of data.
^C^C64 bytes from 93.184.216.34: icmp_seq=1 ttl=43 time=236 ms
```

查看 全国解析的 IP 与您服务器 ping 解析的 IP 是否一致。

如不一致, 请检查本文档 (1, 2) 所提到的情况, 再尝试。或请联系我司客服确认我司平台在用 IP, 从而确认在您服务器解析的 IP 与我司平台在用 IP 是否一致。

### 4. Linux 服务器频繁调用域名解析及 DNS 缓存问题

Linux 系统默认没有 DNS 缓存, 想使用 DNS 缓存的话需要自己安装一个服务程序 NSCD(name service cache daemon)。如果 DNS 设置成稳定知名的 DNS 服务器, 一般情况下, 请求时间与安装缓存的差异并不明显。

**注：安装 DNS 缓存服务有以下优缺点：**

**优点：** 对于频繁调用解析的域名，可直接调用系统 DNS 缓存，提高解析速度。

**缺点：** 我司 DNS 解析变改 IP 后，由于缓存服务，您的服务器有缓存，会有 DNS 更新延迟，仍然解析成旧的 IP 地址，手工执行 `nscd -i hosts` 命令或重启该服务 `service nscd restart` 才可以立刻使缓存失效更新。

CentOS 系统安装方法：

```
# yum install nscd
```

Ubuntu / Debian 系统安装方法：

```
apt-get install nscd
```

确认配置已开启 DNS 缓存

```
# vim /etc/nscd.conf
```

```
enable-cache          hosts          yes
```

```
# service nscd start
```

配置参考链接：

<http://linux.die.net/man/5/nscd.conf>

<http://my.oschina.net/phptiger86/blog/138507>

<http://www.36lway.com/linux-nscd-dns-cache/4265.html>