|  |
| --- |
| **蝶信通** |
| 短信（网关）平台接口说明v4.0 |

**目 录**

[**第一章** **概述** 4](#_Toc430596312)

[1.1 协议说明 4](#_Toc430596313)

[1.2 适用范围 4](#_Toc430596314)

[1.3 参考资料 4](#_Toc430596315)

[1.4 缩略语 4](#_Toc430596316)

[**第二章** **通信方式** 4](#_Toc430596317)

[**第三章 协议报文定义** 5](#_Toc430596318)

[3.1 报文域属性说明 5](#_Toc430596319)

[3.2 消息报文定义 5](#_Toc430596320)

[3.2.1 下行短信提交MTSMS 5](#_Toc430596321)

[3.2.2 查询当前预付费用户余额QUERYAMTF 7](#_Toc430596322)

[3.2.3 上行URL验证MOURLVERIFY 8](#_Toc430596323)

[3.2.4 上行短信推送SMSPUSH 9](#_Toc430596324)

[3.2.5 上行状态报告推送 SMSRPTPUSH 10](#_Toc430596325)

[**第四章 附录-码表** 11](#_Toc430596326)

[4.1 认证返回码authstatus 11](#_Toc430596327)

[4.2 下行短信提交响应码mtrespcode 11](#_Toc430596328)

[4.3 查询余额响应码queryamtfrespcode 12](#_Toc430596329)

[4.4 上行接收响应码morespcode 12](#_Toc430596330)

[4.5 状态报告Stat 12](#_Toc430596331)

[**第五章 接口调用示例代码** 13](#_Toc430596332)

[5.1 .Net示例代码 13](#_Toc430596333)

[5.1.1 提交短信mt 13](#_Toc430596334)

[5.1.2 查询当前预付费用户余额QUERYAMTF 17](#_Toc430596335)

[5.1.3 上行URL验证MOURLVERIFY 19](#_Toc430596336)

[5.1.4 上行短信推送SMSMOPUSH 19](#_Toc430596337)

[5.1.5 上行状态报告推送 SMSRPTPUSH 21](#_Toc430596338)

[5.2 Java示例代码 23](#_Toc430596339)

[5.2.1 提交短信mt 23](#_Toc430596340)

[5.2.2 查询当前预付费用户余额QUERYAMTF 23](#_Toc430596341)

[5.2.3 上行URL验证MOURLVERIFY 23](#_Toc430596342)

[5.2.4 上行短信推送SMSMOPUSH 23](#_Toc430596343)

[5.2.5 上行状态报告推送 SMSRPTPUSH 23](#_Toc430596344)

[5.3 其他示例代码 32](#_Toc430596345)

### 第一章 概述

### 1.1协议说明

本短信API是使用HTTP并遵循REST原则设计的Web服务接口，可以使用几乎任何客户端和任何编程语言与REST API进行交互。通过发送简单的HTTP POST请求就可以轻松接入使用。

### 1.2适用范围

### 1.3参考资料

HTTP 1.0

### 1.4缩略语

|  |  |  |
| --- | --- | --- |
| **缩写** | **全称** | **说明** |
|  |  |  |

### 第二章 通信方式



通信支持HTTP / HTTPS。

### 第三章 协议报文定义

### 3.1报文域属性说明

元素约束（出现次数）符号说明：

|  |  |
| --- | --- |
| **符号** | **说明** |
| ? | 0..1，可选项 |
| \* | 0..n，可以没有，也可以有多项 |
| + | 1..n，至少有1项，也可以有多项 |
| 1 | 必须有且只有1项 |

### 3.2消息报文定义

**3.2.1 下行短信提交MTSMS**

|  |  |  |
| --- | --- | --- |
| **HTTP方法** | **URL** | **描述** |
| POST | ${baseurl}/mtsms/${account}/${token} | 下行短信提交请求URL  baseurl:服务商提供基础URL  account:用户名  token:用户鉴权串， account+时间戳yyyyMMddHHmmss+password做md5编码(Hex字符串) |

|  |  |  |
| --- | --- | --- |
| **报头** | **约束** | **描述** |
| Accept | 1 | application/json |
| Content-Type | 1 | application/json;charset=utf-8 |
| Authorization | 1 | Basic用户ID+“:”+时间戳yyyyMMddHHmmss， 使用Base64编码 |

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **类型** | **描述** |
| Cli\_Msg\_Id | ? | String(24) | 客户流水号，可在响应中携带返回，最长24位，数字、字母 |
| Pk\_total | ? | Unsigned Integer | 相同信息总条数，从1开始  默认为1 |
| Pk\_number | ? | Unsigned Integer | 相同信息序号，从1开始  默认为1 |
| Registered\_Delivery | ? | Unsigned Integer | 是否要求返回状态确认报告：  0：不需要  1：需要  默认为0-不要状态报告 |
| Msg\_level | ? | Unsigned Integer | 信息级别 （0-9）数字越大，级别越高  默认为0 |
| Service\_Id | ? | String | 业务类型，是数字、字母和符号的组合  默认为空 |
| TP\_pId | ? | Unsigned Integer | GSM协议类型。详细是解释请参考GSM03.40中的9.2.3.9  默认0 |
| TP\_udhi | ? | Unsigned Integer | GSM协议类型。详细是解释请参考GSM03.40中的9.2.3.23,仅使用1位  默认0 |
| Msg\_Fmt | ? | Unsigned Integer | 短信内容编码：  0：ASCII串  3：短信写卡操作  4：二进制信息  8：UCS2编码  15：含GB汉字  默认为15 |
| Msg\_src | ? | String | 信息内容来源(数字、英文)  默认为空 |
| Src\_Id | ? | String | 源号码，子扩展号，如可扩展，则扩展在短信平台分配的扩展号后，但总号码不超过21位  默认为空-不扩展、使用短信平台分配的父扩展号 |
| Dest\_terminal\_Id | 1 | String[] | 手机号码（最大21位），集合表示。  单次提交最多不能超过客户带宽。  手机号建议不重复，不强制限制。 |
| Msg\_Content | 1 | Byte[] | 短信内容，使用Msg\_Fmt编码编码为Byte[] |

响应消息：

|  |  |  |
| --- | --- | --- |
| **HTTP响应码** | **响应短语** | **描述** |
| 200/401 | OK/Unauthorized | 接收完成/鉴权失败 |

下发接收完成时（200 OK）：

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **类型** | **描述** |
| Rets | 1 | Object[] | 结果返回集合，每个提交手机号对应以下子元素 |
| Rspcode | 子元素  + | Integer | 下行响应码  取值见附录- [**mtrespcode**](#_下行短信提交响应码mtrespcode) |
| Msg\_Id | 子元素  + | String | 信息标识，用于对应状态报告  用户提供流水号时：系统流水号+“-”+用户流水号  用户未提供流水号时：系统流水号  响应码不成功时无此内容 |
| Dest\_terminal\_Id | 子元素  + | String | 手机号 |

鉴权失败时（401 Unauthorized）：

|  |  |  |  |
| --- | --- | --- | --- |
| **报头** | **约束** | **描述** | |
| WWW-Authenticate | 1 | Basic realm= Unauthorized -status:${authstatus}  authstatus:取值见附录- **authstatus** | |
| **消息体** | **约束** | **值** | **描述** |
|  |  |  |  |

**3.2.2查询当前预付费用户余额QUERYAMTF**

|  |  |  |
| --- | --- | --- |
| **HTTP方法** | **URL** | **描述** |
| POST | ${baseurl}/queryamtf/${account}/${token} | 下行短信提交请求URL  baseurl:服务商提供基础URL  account:用户名  token:用户鉴权串， account+时间戳yyyyMMddHHmmss+password做md5编码(Hex字符串) |

|  |  |  |
| --- | --- | --- |
| **报头** | **约束** | **描述** |
| Accept | 1 | application/json |
| Content-Type | 1 | application/json;charset=utf-8 |
| Authorization | 1 | Basic 用户名: 时间戳yyyyMMddHHmmss 使用Base64编码 |

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **类型** | **描述** |
|  |  |  |  |

响应消息：

|  |  |  |
| --- | --- | --- |
| **HTTP响应码** | **响应短语** | **描述** |
| 200/401 | OK/Unauthorized | 接收完成/鉴权失败 |

下发接收完成时（200 OK）：

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **类型** | **描述** |
| Rspcode | 1 | Integer | 查询余额响应码  取值见附录-[**queryamtfrespcode**](#_查询余额响应码queryamtfrespcode) |
| Count | ？ | Integer | 当前用户所属账户余额（响应吗为成功时存在） |

鉴权失败时（401 Unauthorized）：

|  |  |  |
| --- | --- | --- |
| **报头** | **约束** | **描述** |
| WWW-Authenticate | 1 | Basic realm= Unauthorized -status:${authstatus}  authstatus:取值见附录- [**authstatus**](#_认证返回码authstatus) |

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **值** | **描述** |
|  |  |  |  |

**3.2.3 上行URL验证MOURLVERIFY**

|  |  |  |
| --- | --- | --- |
| **HTTP方法** | **URL** | **描述** |
| POST | 客户提供上行URL |  |

|  |  |  |
| --- | --- | --- |
| **报头** | **约束** | **描述** |
| Accept | 1 | application/json |
| Content-Type | 1 | application/json;charset=utf-8 |

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **类型** | **描述** |
| Cmd | 1 | String | 标识，内容固定为Test |

响应消息：

|  |  |  |
| --- | --- | --- |
| **HTTP响应码** | **响应短语** | **描述** |
| 200 | OK | 接收完成 |

接收完成时（200 OK）：

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **类型** | **描述** |
| Ret | 1 | String | 固定返回0 |

**3.2.4 上行短信推送SMSPUSH**

|  |  |  |
| --- | --- | --- |
| **HTTP方法** | **URL** | **描述** |
| POST | ${客户提供上行URL} /smsmopush |  |

|  |  |  |
| --- | --- | --- |
| **报头** | **约束** | **描述** |
| Accept | 1 | application/json |
| Content-Type | 1 | application/json;charset=utf-8 |

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **类型** | **描述** |
| Msg\_Id | 1 | String | 信息标识 |
| Dest\_Id | 1 | String | 用户上行服务号 |
| TP\_pId | 1 | Unsigned Integer | GSM协议类型。详细是解释请参考GSM03.40中的9.2.3.9  默认0 |
| TP\_udhi | 1 | Unsigned Integer | GSM协议类型。详细是解释请参考GSM03.40中的9.2.3.23,仅使用1位，右对齐  默认0 |
| Msg\_Fmt | 1 | Unsigned Integer | 短信内容编码：  0：ASCII串  3：短信写卡操作  4：二进制信息  8：UCS2编码  15：含GB汉字  默认为15 |
| Src\_terminal\_Id | 1 | String | 用户手机号 |
| Msg\_Content | 1 | Byte[] | 短信内容，使用Msg\_Fmt编码编码为Byte[] |

响应消息：

|  |  |  |
| --- | --- | --- |
| **HTTP响应码** | **响应短语** | **描述** |
| 200 | OK | 接收完成 |

接收完成时（200 OK）：

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **类型** | **描述** |
| Rspcode | 1 | Integer | 上行接收响应码  取值见附录-[**morespcode**](#_上行接收响应码morespcode) |

**3.2.5 上行状态报告推送 SMSRPTPUSH**

|  |  |  |
| --- | --- | --- |
| **HTTP方法** | **URL** | **描述** |
| POST | ${客户提供上行URL} /smsrptpush |  |

|  |  |  |
| --- | --- | --- |
| **报头** | **约束** | **描述** |
| Accept | 1 | application/json |
| Content-Type | 1 | application/json;charset=utf-8 |

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **类型** | **描述** |
| Msg\_Id | 1 | String | 信息标识 对应响应中的msgid |
| Dest\_Id | 1 | String | 服务号 |
| Src\_terminal\_Id | 1 | String | 用户手机号 |
| Stat | 1 | String | 发送短信的应答结果  取值见附录-[**Stat**](#_状态报告Stat) |

响应消息：

|  |  |  |
| --- | --- | --- |
| **HTTP响应码** | **响应短语** | **描述** |
| 200 | OK | 接收完成 |

接收完成时（200 OK）：

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **类型** | **描述** |
| Rspcode | 1 | Integer | 上行接收响应码  取值见附录- [**morespcode**](#_上行接收响应码morespcode) | |

### 第四章 附录-码表

### 4.1认证返回码authstatus

|  |  |
| --- | --- |
| **代码** | **说明** |
| 1 | 系统忙 |
| 3 | 用户鉴权错误 |
| 7 | IP鉴权错误 |
| 9 | 未知错误 |

### 4.2下行短信提交响应码mtrespcode

|  |  |
| --- | --- |
| **代码** | **说明** |
| 0 | 成功 |
| 1 | 手机号码在黑名单 |
| 2 | 手机号码不在白名单 |
| 3 | 短信内容中包含黑名单关键字 |
| 4 | 手机号码未找到相应运营商配置 |
| 5 | 手机号码格式错误 |
| 7 | 下发队列满 需客户重发 |
| 8 | 发送超速 需客户重发 |
| 9 | 未知错误 |
| 10 | 短信内容超长 |
| 11 | 预付费客户余额不足 |
| 12 | 含有未报备关键字 |
| 13 | 下行消息携带号码超过限制（默认为商户1s最大带宽） |

### 4.3查询余额响应码queryamtfrespcode

|  |  |
| --- | --- |
| **代码** | **说明** |
| 0 | 查询成功 |
| 1 | 当前用户属于后付费客户 |
| 2 | 当前用户未分配账户 |
| 9 | 未知错误 |

### 4.4上行接收响应码morespcode

|  |  |
| --- | --- |
| **代码** | **说明** |
| 0 | 成功 |
| 1 | 失败 |

### 4.5状态报告Stat

|  |  |
| --- | --- |
| **代码** | **说明** |
| DELIVRD | 成功 |
| EXPIRED | Message validity period has  expired |
| DELETED | Message has been deleted. |
| UNDELIV | Message is undeliverable |
| ACCEPTD | Message is in accepted state(i.e. has been manually read on behalf of the subscriber by customer service) |
| UNKNOWN | Message is in invalid state |
| REJECTD | Message is in a rejected state |
| MA:xxxx | SMSC不返回响应消息时的状态报告 |
| MB:xxxx | SMSC返回错误响应消息时的状态报告 |
| CA:xxxx | SCP不返回响应消息时的状态报告 |
| CB:xxxx | SCP返回错误响应消息时的状态报告 |
| YC:0000 | 人工审核错误 |
| YC:00XX | XX对应下行短信提交响应码mtrespcode，表示系统内部模块处理失败返回码。  客户侧处理方式同下行短信提交响应码。 |

### 第五章 接口调用示例代码

### 5.1 .Net示例代码

**5.1.2 提交短信mt**

using System;

using System.Collections.Generic;

using System.IO;

using System.Net;

using System.Text;

namespace WebRequestStandardDemo

{

public class MtStandardDemo

{

//基础URL，由平台提供 http://{IP}:{port}/{version}

private const string BaseUrl = "http://123.57.48.46:28080/chif10";

//方法，由平台提供

private const string Method = "mtsms";

//用户ID

private const string UserId = "http05";

//帐号密码

private const string Password = "\*\*\*\*\*\*";

//客户流水号，可包含数字字母 （可以不填）

private const string CliMsgId = "msg00001";

//相同信息总条数，从 1开,默认为 1

private const int PkTotal = 1;

//相同信息序号，从 1开始,默认为 1

private const int PkNumber = 1;

//是否要求返回状态确认报告：0：不需要 1：需要 默认为 0-不要状态报告

private const int RegisteredDelivery = 0;

//信息级别 （0-9）数字越大，级别越高 默认为 0

private const int MsgLevel = 0;

//业务类型，是数字、字母和符号的组合 默认为空

private const string ServiceId = "";

//GSM 协议类型。默认 0

private const int TPPId = 0;

//GSM 协议类型 仅使用 1 位 默认 0

private const int TPUdhi = 0;

//短信内容编码： 0：ASCII 串 3：短信写卡操作 4：二进制信息 8：UCS2 编码 15：含 GB 汉字 默认为 15

private const int MsgFmt = 15;

//信息内容来源(数字、英文) 默认为空

private const string MsgSrc = "";

//源号码，子扩展号，如可扩展，则扩展在短信平台分配的扩展号后，但总号码不超过 21 位 默认为空-不扩展、使用短信平台分配的父扩展号

private const string SrcId = "";

//手机号码（最大 21 位），集合表示。单次提交最多不能超过客户带宽。 手机号建议不重复，不强制限制。

private static string[] DestTerminalId = { "15024379262", "13466566405" };

//短信内容，使用 Msg\_Fmt 编码编码为 Byte[]

private const string MsgContent = "测试";

/// <summary>

/// 调用平台短信接口

/// </summary>

/// <returns>返回结果</returns>

public static string CallApi()

{

//拼接Token，account+时间戳yyyyMMddHHmmss+password 做 md5 编码(Hex 字符串)

string timeStamp = DateTime.Now.ToString("yyyyMMddHHmmss");

string token = string.Format("{0}{1}{2}", UserId,timeStamp, Password);

//拼接URL

string url = string.Format("{0}/{1}/{2}/{3}", BaseUrl, Method, UserId, ToolHelper.GetMd5(token));

Dictionary<string, object> postData = new Dictionary<string, object>(); //POST参数

postData.Add("Cli\_Msg\_Id", CliMsgId); //参数：客户流水号

postData.Add("Pk\_total ", PkTotal); //参数：相同信息总条数

postData.Add("Pk\_number", PkNumber); //参数：相同信息序号

postData.Add("Registered\_Delivery", RegisteredDelivery); //参数：是否要求返回状态确认报告

postData.Add("Msg\_level", MsgLevel); //参数：信息级别 （0-9）数字越大，级别越高

postData.Add("Service\_Id", ServiceId); //参数：业务类型，是数字、字母和符号的组合

postData.Add("TP\_pId", TPPId); //参数：GSM 协议类型

postData.Add("TP\_udhi", TPUdhi); //参数：GSM 协议类型

postData.Add("Msg\_Fmt", MsgFmt); //参数：短信内容编码

postData.Add("Msg\_src", MsgSrc); //参数：信息内容来源(数字、英文)

postData.Add("Src\_Id", SrcId); //源号码，子扩展号

postData.Add("Dest\_terminal\_Id", DestTerminalId); //手机号码（最大 21 位），集合表示。

byte[] msgContentByte = System.Text.Encoding.GetEncoding(936).GetBytes(MsgContent); //使用GB2312 转码

postData.Add("Msg\_Content", msgContentByte); //短信内容，使用 Msg\_Fmt 编码编码为 Byte[]。

//手动拼接JSON数据（此处可以使用JSON的序列化工具）

StringBuilder param = new StringBuilder();

param.Append("{");

foreach (var data in postData)

{

if (data.Value.GetType().Name == "String[]" )

{

param.Append("\"" + data.Key + "\"");

param.Append(":");

param.Append("[");

foreach (string str in (string[])(data.Value))

{

param.Append(str + ",");

}

param.Remove(param.Length - 1, 1); //去除末尾的逗号

param.Append("],");

}

else if (data.Value.GetType().Name == "Byte[]")

{

param.Append("\"" + data.Key + "\"");

param.Append(":");

param.Append("[");

foreach (byte by in (byte[])(data.Value))

{

param.Append(by + ",");

}

param.Remove(param.Length - 1, 1); //去除末尾的逗号

param.Append("],");

}

else

{

param.Append(data.Key + ":\"" + data.Value + "\",");

}

}

param.Remove(param.Length - 1, 1); //去除末尾的逗号

param.Append("}");

HttpWebRequest myRequest = (HttpWebRequest)WebRequest.Create(url); //建立Request请求

myRequest.Method = "POST"; //采用POST方式提交

myRequest.Accept = "application/json"; //客户端响应接收数据格式

myRequest.ContentType = "application/json;charset=utf-8;"; //类型

string authorization = ToolHelper.GetBase64(UserId + ":" + timeStamp); //Base64加密

myRequest.Headers.Add("Authorization", authorization); //用户鉴权

UTF8Encoding encoding = new UTF8Encoding(); //参数编码格式

byte[] postParams = encoding.GetBytes(param.ToString()); //转化编码格式

myRequest.ContentLength = postParams.Length; //内容长度

Stream postStream = myRequest.GetRequestStream(); //请求流数据

//发送数据

postStream.Write(postParams, 0, postParams.Length);

postStream.Flush();

postStream.Close();

string result = string.Empty;

try

{

HttpWebResponse myResponse = (HttpWebResponse)myRequest.GetResponse(); //获取

if (myResponse.StatusCode == HttpStatusCode.OK) //返回正确（200 OK）

{

StreamReader reader = new StreamReader(myResponse.GetResponseStream(), Encoding.UTF8); //读取返回结果

result = reader.ReadToEnd(); //获取JSON数据

}

}

catch (Exception ex)

{

result = ex.Message; //异常结果

}

return result;

}

}

}

**5.1.2 查询当前预付费用户余额QUERYAMTF**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

namespace WebRequestStandardDemo

{

public class QueryAmtStandardDemo

{

//基础URL，由平台提供 http://{IP}:{port}/{version}

private const string BaseUrl = "http://123.57.48.46:28080/chif10";

//方法，由平台提供

private const string Method = "queryamtf";

//用户ID

private const string UserId = "http05";

//帐号密码

private const string Password = "\*\*\*\*\*\*";

/// <summary>

/// 查询当前预付费用户余额

/// </summary>

/// <returns>返回结果</returns>

public static string CallApi()

{

//拼接Token，account+时间戳yyyyMMddHHmmss+password 做 md5 编码(Hex 字符串)

string timeStamp = DateTime.Now.ToString("yyyyMMddHHmmss");

string token = string.Format("{0}{1}{2}", UserId, timeStamp, Password);

//拼接URL

string url = string.Format("{0}/{1}/{2}/{3}", BaseUrl, Method, UserId, ToolHelper.GetMd5(token));

HttpWebRequest myRequest = (HttpWebRequest)WebRequest.Create(url); //建立Request请求

myRequest.Method = "POST"; //采用POST方式提交

myRequest.Accept = "application/json"; //客户端响应接收数据格式

myRequest.ContentType = "application/json;charset=utf-8;"; //类型

string authorization = ToolHelper.GetBase64(UserId + ":" + timeStamp); //Base64加密

myRequest.Headers.Add("Authorization", authorization); //用户鉴权

string result = string.Empty;

try

{

HttpWebResponse myResponse = (HttpWebResponse)myRequest.GetResponse(); //获取

if (myResponse.StatusCode == HttpStatusCode.OK) //返回正确（200 OK）

{

StreamReader reader = new StreamReader(myResponse.GetResponseStream(), Encoding.UTF8); //读取返回结果

result = reader.ReadToEnd(); //获取JSON数据

}

}

catch (Exception ex)

{

result = ex.Message; //异常结果

}

return result;

}

}

}

**5.1.3 上行URL验证MOURLVERIFY**

using System.Net.Http;

using System.Web.Http;

using WebRestfulStandard.Models;

namespace WebRestfulStandard.Controllers

{

public class UrlVerifyController : ApiController

{

/// <summary>

/// 上行 URL 验证

/// </summary>

/// <param name="model">传送的Model对象</param>

/// <returns>Json数据</returns>

public HttpResponseMessage Post(UrlVerifyModel model)

{

string json = string.Empty;

//验证参数

if (model.Cmd == null)

{

json = "{\"Ret\":\"1\"}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

//需要编写的逻辑，这里是例子（数据库操作等在此编写）

FileWriteHelper.Write("Log\_UrlVerifyStandard.txt", model.ToString());

json = "{\"Ret\":\"0\"}";

//返回结果，正确：{\"Ret\":\"0\"}，错误：{\"Ret\":\"1\"}

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

}

}

**5.1.4 上行短信推送SMSMOPUSH**

using System.Net.Http;

using System.Web.Http;

using WebRestfulStandard;

using WebRestfulStandard.Models;

namespace WebRestfulStandard.Controllers

{

public class SmsMoPushController : ApiController

{

/// <summary>

/// 上行短信推送

/// </summary>

/// <param name="model">传送的Model对象</param>

/// <returns>Json数据</returns>

public HttpResponseMessage Post(SmsMoPushModel model)

{

string json = string.Empty;

//验证参数

if (model.Msg\_Id == null)

{

//信息标识 不存在

json = "{\"Rspcode\":1}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

if (model.Dest\_Id == null)

{

//用户上行服务号 不存在

json = "{\"Rspcode\":1}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

if (model.TP\_pId == null)

{

//GSM 协议类型 不存在

json = "{\"Rspcode\":1}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

if (model.TP\_udhi == null)

{

//GSM 协议类型,仅使用 1位 不存在

json = "{\"Rspcode\":1}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

if (model.Msg\_Fmt == null)

{

//短信内容编码 不存在

json = "{\"Rspcode\":1}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

if (model.Src\_terminal\_Id == null)

{

//用户手机号 不存在

json = "{\"Rspcode\":1}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

if (model.Msg\_Content == null)

{

//短信内容 不存在

json = "{\"Rspcode\":1}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

//需要编写的逻辑，这里是例子（数据库操作等在此编写）

FileWriteHelper.Write("Log\_SmsMoPushStandard.txt",model.ToString());

json = "{\"Rspcode\":0}";

//返回结果，正确：{\"Rspcode\":0}，错误：{\"Rspcode\":1}

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

}

}

**5.1.5上行状态报告推送 SMSRPTPUSH**

using System.Net.Http;

using System.Web.Http;

using WebRestfulStandard;

using WebRestfulStandard.Models;

namespace WebRestfulStandard.Controllers

{

public class SmsRptPushController : ApiController

{

/// <summary>

/// 上行状态报告推送 SMSRPTPUSH

/// </summary>

/// <param name="model">传送的Model对象</param>

/// <returns>Json数据</returns>

public HttpResponseMessage Post(SmsRptPushModel model)

{

string json = string.Empty;

//验证参数

if (model.Msg\_Id == null)

{

//信息标识 不存在

json = "{\"Rspcode\":1}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

if (model.Dest\_Id == null)

{

//用户上行服务号 不存在

json = "{\"Rspcode\":1}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

if (model.Src\_terminal\_Id == null)

{

//用户手机号 不存在

json = "{\"Rspcode\":1}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

if (model.Stat == null)

{

//短信内容 不存在

json = "{\"Rspcode\":1}";

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

//需要编写的逻辑,这里是列子（数据库操作等在此编写）

FileWriteHelper.Write("Log\_SmsRptPushStandard.txt", model.ToString());

json = "{\"Rspcode\":0}";

//返回结果，正确：{\"Rspcode\":0}，错误：{\"Rspcode\":1}

return new HttpResponseMessage { Content = new StringContent(json, System.Text.Encoding.UTF8, "application/json") };

}

}

}

### 5.2 Java示例代码

### 5.1.1提交短信mt

### 5.1.2查询当前预付费用户余额QUERYAMTF

### 5.1.3上行URL验证MOURLVERIFY

### 5.1.4上行短信推送SMSMOPUSH

### 5.1.5上行状态报告推送 SMSRPTPUSH

<%@page **import**=**"java.util.LinkedHashMap"**%>  
<%@page **import**=**"java.util.Map.Entry"**%>  
<%@page **import**=**"java.util.Iterator"**%>  
<%@page **import**=**"java.util.UUID"**%>  
<%@page **import**=**"java.util.HashMap"**%>  
<%@page **import**=**"java.util.Map"**%>  
<%@page **import**=**"java.io.IOException"**%>  
<%@page **import**=**"java.io.InputStreamReader"**%>  
<%@page **import**=**"java.net.URL"**%>  
<%@page **import**=**"java.net.HttpURLConnection"**%>  
<%@page **import**=**"java.io.BufferedReader"**%>  
<%@page **import**=**"java.io.PrintWriter"**%>  
<%@ page language=**"java"** contentType=**"text/html; charset=UTF-8"** pageEncoding=**"UTF-8"**%>  
<%!*//平台Base URL  
// 由平台提供 http://{IP}:{port}/{version}  
// private static final String baseUrlString =  
// "http://121.41.85.249:28080/HIF12";***private static final** String baseUrl = **"http://123.57.48.46:28080/HIF12"**;  
**private static final** String enterpriseUrl = **"http://localhost:8080/yuecheng/rest/simple"**;  
*// 用户ID***private static final** String userId = **"http05"**;  
*// 帐号密码***private static final** String password = **"\*\*\*\*\*\*"**;  
*// 短信接收端手机号码集合，用半角逗号（英文逗号）分开，每批发 送的手机号数量不得超过不能超过客户设置带宽。  
// 通常以20个号码做为上限。  
// 手机号建议不重复，不强制限制***private static final** String mobile = **"13466566405"**;  
*// 短信内容，UTF-8 编码字符串，单条通常为 65 汉字以内（根据签 名规则不同），超过限制字数会被分拆，  
// 同时计费条数会根据最终拆 分条数计算，具体由平台拆分结果确定。***private static final** String content = **"确定"**;  
  
*/\*\*  
 \* 查询当前预付费用户余额  
 \*  
 \** ***@return*** *\*/***public static** String postQueryamtf() {  
 PrintWriter printWriter = **null**;  
 BufferedReader bufferedReader = **null**;  
 String responseResult = **new** String();  
 HttpURLConnection httpURLConnection = **null**;  
  
 **try** {  
 URL realUrl = **new** URL(baseUrl + **"/queryamtf/"** + userId);  
 *// 打开和URL之间的连接* httpURLConnection = (HttpURLConnection) realUrl.openConnection();  
 *// 设置通用的请求属性* httpURLConnection.setRequestProperty(**"accept"**, **"application/json"**);  
 httpURLConnection.setRequestProperty(**"Content-Type"**, **"application/json;charset=utf-8"**);  
 httpURLConnection.setDoOutput(**true**);  
 httpURLConnection.setDoInput(**true**);  
 *// 获取URLConnection对象对应的输出流* printWriter = **new** PrintWriter(httpURLConnection.getOutputStream());  
 *// flush输出流的缓冲* printWriter.flush();  
 *// 根据ResponseCode判断连接是否成功* **int** responseCode = httpURLConnection.getResponseCode();  
 **if** (responseCode != 200) {  
 responseResult = **" Error==="** + responseCode;  
 } **else** {  
 *// out.println("Post Success!");* }  
 *// 定义BufferedReader输入流来读取URL的ResponseData* bufferedReader = **new** BufferedReader(**new** InputStreamReader(httpURLConnection.getInputStream()));  
 String line;  
 **while** ((line = bufferedReader.readLine()) != **null**) {  
 responseResult += line;  
 }  
 } **catch** (Exception e) {  
 responseResult = **"send post request error!"** + e;  
 } **finally** {  
 httpURLConnection.disconnect();  
 **try** {  
 **if** (printWriter != **null**) {  
 printWriter.close();  
 }  
 **if** (bufferedReader != **null**) {  
 bufferedReader.close();  
 }  
 } **catch** (IOException ex) {  
 ex.printStackTrace();  
 }  
  
 }  
 **return** responseResult;  
 }  
  
*/\*\*  
 \* 提交短信  
 \*  
 \** ***@return*** *\*/***public static** String postMt() {  
 PrintWriter printWriter = **null**;  
 BufferedReader bufferedReader = **null**;  
 String responseResult = **""**;  
 StringBuffer params = **new** StringBuffer();  
 HttpURLConnection httpURLConnection = **null**;  
  
 Map<String, Object> requestParamsMap = **new** HashMap<String, Object>();  
 requestParamsMap.put(**"Userid"**, userId);  
 requestParamsMap.put(**"Passwd"**, password);  
 requestParamsMap.put(**"Cli\_Msg\_Id"**, UUID.randomUUID());  
 requestParamsMap.put(**"Mobile"**, mobile);  
 requestParamsMap.put(**"Content"**, content);  
  
 Iterator<Entry<String, Object>> it = requestParamsMap.entrySet().iterator();  
 params.append(**"{"**);  
 **while** (it.hasNext()) {  
 Map.Entry element = (Map.Entry) it.next();  
 params.append(**"\""** + element.getKey() + **"\""** + **":\""** + element.getValue() + **"\","**);  
 }  
 params.deleteCharAt(params.lastIndexOf(**","**));  
 params.append(**"}"**);  
  
 **try** {  
 URL realUrl = **new** URL(baseUrl + **"/mt"**);  
 *// 打开和URL之间的连接* httpURLConnection = (HttpURLConnection) realUrl.openConnection();  
 *// 设置通用的请求属性* httpURLConnection.setRequestProperty(**"accept"**, **"application/json"**);  
 httpURLConnection.setRequestProperty(**"Content-Type"**, **"application/json;charset=utf-8"**);  
 httpURLConnection.setRequestProperty(**"Content-Length"**, String.valueOf(params.length()));  
 httpURLConnection.setDoOutput(**true**);  
 httpURLConnection.setDoInput(**true**);  
 *// 获取URLConnection对象对应的输出流* printWriter = **new** PrintWriter(httpURLConnection.getOutputStream());  
 *// 发送请求参数* printWriter.write(params.toString());  
 *// flush输出流的缓冲* printWriter.flush();  
 *// 根据ResponseCode判断连接是否成功* **int** responseCode = httpURLConnection.getResponseCode();  
 **if** (responseCode != 200) {  
 responseResult = **" Error==="** + responseCode;  
 } **else** {  
*// out.println("Post Success!");* }  
 *// 定义BufferedReader输入流来读取URL的ResponseData* bufferedReader = **new** BufferedReader(**new** InputStreamReader(httpURLConnection.getInputStream()));  
 String line;  
 **while** ((line = bufferedReader.readLine()) != **null**) {  
 responseResult += line;  
 }  
 } **catch** (Exception e) {  
 responseResult = **"send post request error!"** + e;  
 } **finally** {  
 httpURLConnection.disconnect();  
 **try** {  
 **if** (printWriter != **null**) {  
 printWriter.close();  
 }  
 **if** (bufferedReader != **null**) {  
 bufferedReader.close();  
 }  
 } **catch** (IOException ex) {  
 ex.printStackTrace();  
 }  
  
 }  
 **return** responseResult;  
 }  
  
*/\*\*  
 \* 上行URL验证  
 \*  
 \** ***@return*** *\*/***public static** String postMourlverify() {  
 PrintWriter printWriter = **null**;  
 BufferedReader bufferedReader = **null**;  
 String responseResult = **new** String();  
 HttpURLConnection httpURLConnection = **null**;  
  
 **try** {  
 URL realUrl = **new** URL(enterpriseUrl + **"/index.jsp"**);  
 *// 打开和URL之间的连接* httpURLConnection = (HttpURLConnection) realUrl.openConnection();  
 *// 设置通用的请求属性* httpURLConnection.setRequestProperty(**"accept"**, **"application/json"**);  
 httpURLConnection.setRequestProperty(**"Content-Type"**, **"application/json;charset=utf-8"**);  
 httpURLConnection.setDoOutput(**true**);  
 httpURLConnection.setDoInput(**true**);  
 *// 获取URLConnection对象对应的输出流* printWriter = **new** PrintWriter(httpURLConnection.getOutputStream());  
 *// 发送请求参数* printWriter.write(**"{\"Cmd\":\"Test\"}"**);  
 *// flush输出流的缓冲* printWriter.flush();  
 *// 根据ResponseCode判断连接是否成功* **int** responseCode = httpURLConnection.getResponseCode();  
 **if** (responseCode != 200) {  
 responseResult = **" Error==="** + responseCode;  
 } **else** {  
 *// out.println("Post Success!");* }  
 *// 定义BufferedReader输入流来读取URL的ResponseData* bufferedReader = **new** BufferedReader(**new** InputStreamReader(httpURLConnection.getInputStream()));  
 String line;  
 **while** ((line = bufferedReader.readLine()) != **null**) {  
 responseResult += line;  
 }  
 } **catch** (Exception e) {  
 responseResult = **"send post request error!"** + e;  
 } **finally** {  
 httpURLConnection.disconnect();  
 **try** {  
 **if** (printWriter != **null**) {  
 printWriter.close();  
 }  
 **if** (bufferedReader != **null**) {  
 bufferedReader.close();  
 }  
 } **catch** (IOException ex) {  
 ex.printStackTrace();  
 }  
  
 }  
 **return** responseResult;  
 }  
  
*/\*\*  
 \* 上行短信推送  
 \*  
 \** ***@return*** *\*/***public static** String postSmsmopush() {  
 PrintWriter printWriter = **null**;  
 BufferedReader bufferedReader = **null**;  
 String responseResult = **new** String();  
 HttpURLConnection httpURLConnection = **null**;  
 StringBuffer params = **new** StringBuffer();  
  
 Map<String, Object> requestParamsMap = **new** LinkedHashMap<String, Object>();  
 requestParamsMap.put(**"Msg\_Id"**, UUID.randomUUID());*// 信息标识* requestParamsMap.put(**"Dest\_Id"**, **"106901110001"**);*// 用户上行服务号* requestParamsMap.put(**"Mobile"**, mobile);  
 requestParamsMap.put(**"Content"**, content);  
  
 Iterator<Entry<String, Object>> it = requestParamsMap.entrySet().iterator();  
 params.append(**"{"**);  
 **for** (String key : requestParamsMap.keySet()) {  
 params.append(**"\""** + key + **"\""** + **":\""** + requestParamsMap.get(key) + **"\","**);  
 }  
 params.deleteCharAt(params.lastIndexOf(**","**));  
 params.append(**"}"**);  
  
 **try** {  
 URL realUrl = **new** URL(enterpriseUrl + **"/smsmopush.jsp"**);  
 *// 打开和URL之间的连接* httpURLConnection = (HttpURLConnection) realUrl.openConnection();  
 *// 设置通用的请求属性* httpURLConnection.setRequestProperty(**"accept"**, **"application/json"**);  
 httpURLConnection.setRequestProperty(**"Content-Type"**, **"application/json;charset=utf-8"**);  
 httpURLConnection.setDoOutput(**true**);  
 httpURLConnection.setDoInput(**true**);  
 *// 获取URLConnection对象对应的输出流* printWriter = **new** PrintWriter(httpURLConnection.getOutputStream());  
 *// 发送请求参数* printWriter.write(params.toString());  
 *// flush输出流的缓冲* printWriter.flush();  
 *// 根据ResponseCode判断连接是否成功* **int** responseCode = httpURLConnection.getResponseCode();  
 **if** (responseCode != 200) {  
 responseResult = **" Error==="** + responseCode;  
 } **else** {  
*// out.println("Post Success!");* }  
 *// 定义BufferedReader输入流来读取URL的ResponseData* bufferedReader = **new** BufferedReader(**new** InputStreamReader(httpURLConnection.getInputStream()));  
 String line;  
 **while** ((line = bufferedReader.readLine()) != **null**) {  
 responseResult += line;  
 }  
 } **catch** (Exception e) {  
 responseResult = **"send post request error!"** + e;  
 } **finally** {  
 httpURLConnection.disconnect();  
 **try** {  
 **if** (printWriter != **null**) {  
 printWriter.close();  
 }  
 **if** (bufferedReader != **null**) {  
 bufferedReader.close();  
 }  
 } **catch** (IOException ex) {  
 ex.printStackTrace();  
 }  
  
 }  
 **return** responseResult;  
 }  
  
*/\*\*  
 \* 上行状态报告推送   
 \*  
 \** ***@return*** *\*/***public static** String postSmsrptpush() {  
 PrintWriter printWriter = **null**;  
 BufferedReader bufferedReader = **null**;  
 String responseResult = **new** String();  
 HttpURLConnection httpURLConnection = **null**;  
 StringBuffer params = **new** StringBuffer();  
  
 Map<String, Object> requestParamsMap = **new** LinkedHashMap<String, Object>();  
 requestParamsMap.put(**"Msg\_Id"**, UUID.randomUUID());*// 信息标识* requestParamsMap.put(**"Dest\_Id"**, **"106901110001"**);*// 用户上行服务号* requestParamsMap.put(**"Mobile"**, mobile);  
 requestParamsMap.put(**"Status"**, **"DELIVRD"**);  
  
 params.append(**"{"**);  
 **for** (String key : requestParamsMap.keySet()) {  
 params.append(**"\""** + key + **"\""** + **":\""** + requestParamsMap.get(key) + **"\","**);  
 }  
 params.deleteCharAt(params.lastIndexOf(**","**));  
 params.append(**"}"**);  
  
 **try** {  
 URL realUrl = **new** URL(enterpriseUrl + **"/smsrptpush.jsp"**);  
 *// 打开和URL之间的连接* httpURLConnection = (HttpURLConnection) realUrl.openConnection();  
 *// 设置通用的请求属性* httpURLConnection.setRequestProperty(**"accept"**, **"application/json"**);  
 httpURLConnection.setRequestProperty(**"Content-Type"**, **"application/json;charset=utf-8"**);  
 httpURLConnection.setDoOutput(**true**);  
 httpURLConnection.setDoInput(**true**);  
 *// 获取URLConnection对象对应的输出流* printWriter = **new** PrintWriter(httpURLConnection.getOutputStream());  
 *// 发送请求参数* printWriter.write(params.toString());  
 *// flush输出流的缓冲* printWriter.flush();  
 *// 根据ResponseCode判断连接是否成功* **int** responseCode = httpURLConnection.getResponseCode();  
 **if** (responseCode != 200) {  
 responseResult = **" Error==="** + responseCode;  
 } **else** {  
*// out.println("Post Success!");* }  
 *// 定义BufferedReader输入流来读取URL的ResponseData* bufferedReader = **new** BufferedReader(**new** InputStreamReader(httpURLConnection.getInputStream()));  
 String line;  
 **while** ((line = bufferedReader.readLine()) != **null**) {  
 responseResult += line;  
 }  
 } **catch** (Exception e) {  
 responseResult = **"send post request error!"** + e;  
 } **finally** {  
 httpURLConnection.disconnect();  
 **try** {  
 **if** (printWriter != **null**) {  
 printWriter.close();  
 }  
 **if** (bufferedReader != **null**) {  
 bufferedReader.close();  
 }  
 } **catch** (IOException ex) {  
 ex.printStackTrace();  
 }  
  
 }  
 **return** responseResult;  
 }%>  
  
  
<%  
 String m = request.getParameter(**"m"**);  
 **if** (m.equals(**"mt"**)) {  
 out.println(postMt());  
 } **else if**(m.equals(**"queryamtf"**)) {  
 out.println(postQueryamtf());  
 } **else if**(m.equals(**"mourlverify"**)) {  
 out.println(postMourlverify());  
 } **else if**(m.equals(**"smsrptpush"**)) {  
 out.println(postSmsrptpush());  
 } **else if**(m.equals(**"smsmopush"**)) {  
 out.println(postSmsmopush());  
 }  
 %>

## 其他示例代码

## 请向平台供应商索取!