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

**文档变更记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **变更内容说明** | **版本号** | **版本日期** | **执笔人** |
| 1 | 初稿,基本协议，MT、MO、状态报告、查询余额  下行为了兼容cmpp协议的长短信，数据短信，计费，服务代码等功能，字段设置以cmpp协议为蓝本，协议保留cmpp协议的所有功能 | 1.0 | 2014/12/21 |  |
| 2 | 简化版本修订：   1. 抽取协议统一请求格式 2. 增加简化MO、MT 3. 版本修订为HIF1.1 | 1.1 | 2015/3/1 |  |
| 3 | 极度简化版本：   1. 鉴权方式改变，使用参数userid， password 2. MT采用http POST | 1.2 | 2015/3/16 |  |
| 4 | 修改MT的mobile字段，从String[]变为逗号分隔的String，  但MT 响应仍保持Rets数组形式 | 1.2 | 2015/3/19 |  |
| 3 | 正式稿 | 1.1 | 2015/3/16 |  |

**目录**

[**第一章** **概述** 6](#_Toc430596361)

[1.1 协议说明 6](#_Toc430596362)

[1.2 版本说明 6](#_Toc430596363)

[1.3 适用范围 6](#_Toc430596364)

[1.4 参考资料 7](#_Toc430596365)

[1.5 缩略语 7](#_Toc430596366)

[**第二章 通信方式** 7](#_Toc430596367)

[**第三章 协议说明** 7](#_Toc430596368)

[3.1 报文域属性说明 7](#_Toc430596369)

[3.2 EC请求平台（下行）统一格式 8](#_Toc430596370)

[3.2.1 平台Base URL 8](#_Toc430596371)

[3.2.2 业务URL格式 8](#_Toc430596372)

[3.2.3 HTTP标准包头字段 9](#_Toc430596373)

[3.3 平台做客户端请求EC（上行）统一格式 9](#_Toc430596374)

[3.3.1 客户提供上行Base URL 9](#_Toc430596375)

[3.3.2 上行业务完整URL 10](#_Toc430596376)

[3.4 编码 10](#_Toc430596377)

[**第四章 协议报文定义** 10](#_Toc430596378)

[4.1 EC请求平台(下行)协议报文定义 10](#_Toc430596379)

[4.1.1 提交短信mt 10](#_Toc430596380)

[4.1.2 查询当前预付费用户余额QUERYAMTF 13](#_Toc430596381)

[4.2 平台做客户端请求EC（上行）协议报文定义 14](#_Toc430596382)

[4.2.1 上行URL验证MOURLVERIFY 14](#_Toc430596383)

[4.2.2 上行短信推送SMSMOPUSH 15](#_Toc430596384)

[4.2.3 上行状态报告推送 SMSRPTPUSH 16](#_Toc430596385)

[**第五章 附录-码表** 17](#_Toc430596386)

[5.1 认证返回码authstatus 17](#_Toc430596387)

[5.2 下行短信提交响应码mtrespcode 17](#_Toc430596388)

[5.3 查询余额响应码queryamtfrespcode 18](#_Toc430596389)

[5.4 上行接收响应码morespcode 18](#_Toc430596390)

[5.5 状态报告Stat 19](#_Toc430596391)

[**第六章 接口调用示例代码** 19](#_Toc430596392)

[6.1 .Net示例代码 19](#_Toc430596393)

[6.1.1 提交短信mt 19](#_Toc430596394)

[6.1.2 查询当前预付费用户余额QUERYAMTF 22](#_Toc430596395)

[6.1.3 上行URL验证MOURLVERIFY 23](#_Toc430596396)

[6.1.4 上行短信推送SMSMOPUSH 24](#_Toc430596397)

[6.1.5 上行状态报告推送 SMSRPTPUSH 25](#_Toc430596398)

[6.2 .Java示例代码 27](#_Toc430596399)

[6.2.1 提交短信mt 27](#_Toc430596400)

[6.2.2 查询当前预付费用户余额QUERYAMTF 27](#_Toc430596401)

[6.2.3 上行URL验证MOURLVERIFY 27](#_Toc430596402)

[6.2.4 上行短信推送SMSMOPUSH 27](#_Toc430596403)

[6.2.5 上行状态报告推送 SMSRPTPUSH 27](#_Toc430596404)

* 1. [其他示例代码 36](#_Toc430596405)

### 第一章 概述

## 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。

## 版本说明

在作为CMPP变体的http协议chif1.0推出后，为了简化协议开发难度，对协议的下行和上行部分进行相应地修改，形成本简化协议，保留业务逻辑必要的核心字段，去掉扩展功能的若干字段（本接口不再具备发送数据短信能力）。

**chif1.2与chif1.0的异同：**

1. 安全认证方式相同，仍然为报文头携带Authorization信息base64编码，URL携带MD5 token
2. 下行MTSMSSubmit/上行MOSmsPush报文简化
3. 状态报告推送方式保持不变
4. MO/MT短信内容传递方式不使用Byte[],而是使用Base64编码的原始UTF-8字符串进行传递。

## 适用范围

## 参考资料

HTTP 1.0

## 缩略语

|  |  |  |
| --- | --- | --- |
| **缩写** | **全称** | **说明** |
| EC | Enterprise customer | 企业客户 |
| MT | Mobile Terminal | 短信下行 |
| MO | Mobile Original | 短信上行 |

### 第二章 通信方式



通信支持HTTP / HTTPS

其中MT/QUERYAMTF 报文为EC作为client请求平台

MOURLVERIFY/SMSMOPUSH/SMSRPTPUSH 报文为平台作为客户端请求EC

### 第三章 协议说明

## 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 EC请求平台（下行）统一格式

### 3.2.1 平台Base URL

由平台提供

http://{IP}:{port}/{version}

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **约束** | **类型** | **描述** |
| IP | 1 | String | 平台的IP地址或者域名。 |
| port | 0 | Integer | 端口号，  端口号不为80时，与IP用半角:隔开  当端口号为80时，此字段可空 |
| version | 1 | String | 协议版本，  当前协议版本为：  HIF12 |

BaseURL示例：

[http://123.57.48.46:8080/HIF12/](http://192.168.1.1:8080/HIF12/)

<http://api.pro-grouop.com/HIF12/>

### 3.2.2 业务URL格式

注：Base URL与业务URL相拼接为完整请求URL

**帐号鉴权：**

/{funcion}/[Userid]

举例：下发短信息的url为：

http://api.pro-group.com:28080/HIF12/mt

用户user0001查询账户余额的url为：

http://api.pro-group.com:28080/HIF12/ queryamtf/user0001

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **约束** | **类型** | **描述** |
| func | 1 | String | 业务功能 |
| Userid | 0 | String | 用户ID |

### 3.2.3  HTTP标准包头字段

**包头必填字段：**

Accept:application/json;

Content-Type:application/json;charset=utf-8;

Content-Length:length;

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **约束** | **类型** | **描述** |
| Accept | 1 | String | 客户端响应接收数据格式：  application/json |
| Content-Type | 1 | String | 类型：  application/json;charset=utf-8 |
| Content-Length | 1 | String | 内容长度 |

## 3.3平台做客户端请求EC（上行）统一格式

适用于平台作为客户端向EC发送上行、状态报告时的情形

### 3.2.1客户提供上行Base URL

由EC提供用以接收上行消息和状态报告的Base URL

举例：

EC提供上行Base URL为：

[http://123.57.48.46:1459/serv/](http://192.168.1.1:1459/serv/)

### 3.2.2上行业务完整URL

上行各业务都有各自的function，完整的业务URL会发送至：

${客户提供上行BaseURL} /{function}

举例：

EC提供上行BaseURL为：

[http://123.57.48.46:1459/serv.php/](http://192.168.1.1:1459/serv/)

平台上行短信function为smsmopush, 上行报文会发送至：

[http://123.57.48.46:1459/serv.php/smsmopush/](http://192.168.1.1:1459/serv.php/smsmopush/)

平台状态报告function为smsrptpush

[http://123.57.48.46:1459/serv.php/smsrptpush/](http://192.168.1.1:1459/serv.php/smsrptpush/)

## 3.4编码

编码统一使用UTF-8

### 第四章 协议报文定义

## 4.1 EC请求平台(下行)协议报文定义

### 4.1.1提交短信mt

|  |  |  |
| --- | --- | --- |
| **HTTP方法** | **URL** | **描述** |
| POST | ${baseurl}/mt/ | 参见3.2节说明 |

|  |  |  |  |
| --- | --- | --- | --- |
| **消息体** | **约束** | **类型** | **描述** |
| Userid | 1 | String | 用户ID |
| Passwd | 1 | String | **帐号密码：(可以明文密码或使用32位MD5加密)** |
| Cli\_Msg\_Id | ? | String(24) | 客户流水号，可在响应中携带返回，最长24位，可包含数字字母 |
| Mobile | 1 | String | 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。 |
| Content | 1 | String | 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。 |

示例：

curl -X POST \

-H "Accept: application/json" \

-H "Content-Type: application/json;charset=utf-8" \

-d '{"Userid":"test","Passwd":"test","Cli\_Msg\_Id":"msg00001","Mobile":"18600000000,13910000000","Content":"您的验证码为1234 【xxx】"}' \

http://ip:28080/HIF12/mt/

**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 |
| Mobile | 子元素  + | String | 手机号 |

**返回报文示例：**

{

"Rets": [

{

"Rspcode": 0,

"Msg\_Id": "0315164359000001002462",

"Mobile": "18600000000"

},

{

"Rspcode": 0,

"Msg\_Id": "0315164359000001002463",

"Mobile": "13910101010"

}

]

}

**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** |
|  |  |  |  | |

**4.1.2查询当前预付费用户余额QUERYAMTF**

|  |  |  |
| --- | --- | --- |
| **HTTP方法** | **URL** | **描述** |
| POST | ${baseurl}/queryamtf/${ Userid } | 参见3.2节说明 |

|  |  |  |  |
| --- | --- | --- | --- |
| **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） |

**反馈示例：**

{

"Rspcode": "0",

"Count": 100

}

**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** |
|  |  |  |  | |

## 4.2平台做客户端请求EC（上行）协议报文定义

### 4.2.1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 |

### 4.2.2上行短信推送SMSMOPUSH

|  |  |  |
| --- | --- | --- |
| **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 |
| Mobile | 1 | String | 用户手机号 |
| Content | 1 | String | 短信内容，UTF-8编码字符串 |

**报文示例：**

{

"Msg\_Id": "0315164359000001002463",

"Dest\_Id": "106901110001",

"Mobile": "18600000000",

"Content": "谢谢！"

}

**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) |

**4.2.3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 |
| Mobile | 1 | String | 用户手机号 |
| Status | 1 | String | 发送短信的应答结果  取值见附录- Stat |

**报文示例：**

{

"Msg\_Id": "msg00002",

"Dest\_Id": "106901110001",

"Mobile": "18600000000",

"Status": "DELIVRD"

}

**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) |

### 第五章 附录-码表

## 5.1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 |

## 5.2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） |

## 5.3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 |

## 5.4上行接收响应码morespcode

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

## 5.5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。 |

### 第六章 接口调用示例代码

## 6.1 .Net示例代码

**6.1.1提交短信mt**

using System;

using System.Collections.Generic;

using System.IO;

using System.Net;

using System.Text;

namespace WebRequestDemo

{

public class MtDemo

{

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

private const string BaseUrl = "http://123.57.48.46:28080/HIF12";

//方法，由平台提供

private const string Method = "mt";

//用户ID

private const string UserId = "http06";

//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";

//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 const string Mobile = "13466566405,15024379262";

//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 const string Content = "这是一个演习";

/// <summary>

/// 调用平台短信接口

/// </summary>

/// <returns>返回结果</returns>

public static string CallApi()

{

string url = string.Format("{0}/{1}", BaseUrl, Method); //拼接URL

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

postData.Add("Userid", UserId); //参数：用户ID

postData.Add("Passwd", Password); //参数：密码

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

postData.Add("Mobile", Mobile); //参数：手机号

postData.Add("Content", Content); //参数：短信内容

//拼接JSON数据

StringBuilder param=new StringBuilder();

param.Append("{");

foreach (var data in postData)

{

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

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;

}

}

}

**6.1.2查询当前预付费用户余额QUERYAMTF**

using System;

using System.IO;

using System.Net;

using System.Text;

namespace WebRequestDemo

{

public class QueryAmtfDemo

{

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

private const string BaseUrl = "http://123.57.48.46:28080/HIF12";

//方法，由平台提供

private const string Method = "queryamtf";

//用户ID

private const string UserId = "http05";

/// <summary>

/// 调用平台接口

/// </summary>

/// <returns>返回结果</returns>

public static string CallApi()

{

string url = string.Format("{0}/{1}/{2}", BaseUrl, Method, UserId); //拼接地址：基础URL+方法+用户ID

HttpWebRequest myRequest = (HttpWebRequest)WebRequest.Create(url); //需要引入using System.Net

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

myRequest.ContentLength = 0; //内容长度

try

{

string result = string.Empty;

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

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

}

return result;

}

catch (Exception ex)

{

return ex.Message; //未经授权(401 Unauthorized)

}

}

}

}

**6.1.3上行URL验证MOURLVERIFY**

using System.Net.Http;

using System.Web.Http;

using WebRestful.Models;

namespace WebRestful.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\_UrlVerify.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") };

}

}

}

**6.1.4上行短信推送SMSMOPUSH**

using System.Net.Http;

using System.Web.Http;

using WebRestful.Models;

namespace WebRestful.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.Mobile == 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.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\_SmsMoPush.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") };

}

}

}

**6.1.5上行状态报告推送 SMSRPTPUSH**

using System.Net.Http;

using System.Web.Http;

using WebRestful.Models;

namespace WebRestful.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.Mobile == 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.Status == 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\_SmsRptPush.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") };

}

}

}

## 6.2 .Java示例代码

### 6.2.1提交短信mt

### 6.2.2查询当前预付费用户余额QUERYAMTF

### 6.2.3上行URL验证MOURLVERIFY

### 6.2.4上行短信推送SMSMOPUSH

### 6.2.5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/diexin /rest/simple"**;  
*// 用户ID***private static final** String userId = **"http06"**;  
*// 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);  
 *// 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());  
 }  
 %>

## 6.3 其他示例代码

请向平台供应商索取!