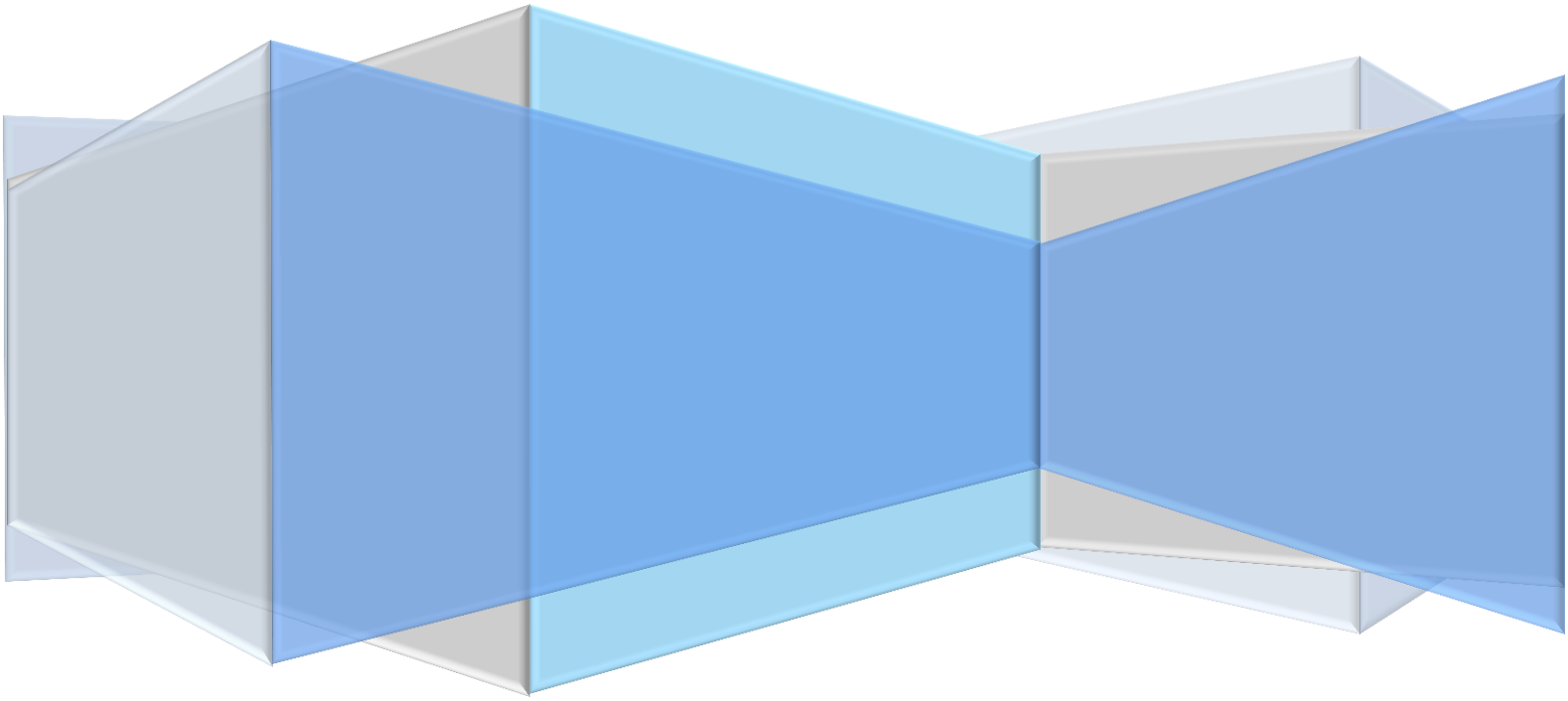

触发接口详细说明文档 V1.8

2020/06/17

重要说明

本人/本公司在使用本软件过程中，将严格遵守国家相关法律、法规、保证本公司信息发布的安全，并切实做到：

（1） 建立健全本公司信息发布的内部保障制度、信息安全保密制度、用户信息安全管理制度，建立健全本公司信息安全责任制度和信息发布的审批制度，严格审查本公司产品所发布信息。

（2） 严格遵守《互联网信息服务管理办法》，对用户编辑的信息内容进行把关，保证信息内容的健康、合法。

（3） 明确本企业的客户群和客户范围，所有通讯受众必须是自愿且明确同意接受企业服务。

（4） 若发送未经用户允许的信息等所造成的影响、投诉等一切损失与本软件版权方无关、由本人/本公司承担全部责任；同时，本人/本公司也应承担若因此给本软件版权方造成的全部损失。

（5） 不利用本软件制作、复制、发布、传播含有下列内容的信息：

- ✧ 反对宪法所确定的基本原则的；
- ✧ 危害国家安全，泄露国家秘密，颠覆国家政权，破坏国家统一的；
- ✧ 损坏国家荣誉和利益的；
- ✧ 煽动民族仇恨、民族歧视，破坏民族团结的；
- ✧ 破坏国家民族宗教政策，宣扬邪教和封建迷信的；
- ✧ 散布谣言，扰乱社会秩序，破坏社会稳定的；
- ✧ 散布淫秽、色情、赌博、暴力、凶杀、恐怖或者教唆犯罪的；
- ✧ 侮辱或者诽谤他人，侵害他人合法权益的；
- ✧ 含有法律、行政法规禁止的其他内容的；

（6） 自信息发布六个月内不修改删除信息发送日志内容，日志记录的备份应至少保持 6 个月以上，在国家机关进行依法查询时，予以提供。

（7） 若发现本公司所发布的信息明显属于上述第（5）款所列内容，保证立即停止传输，并向国家有关机关报告。

（8） 对本公司所发布的信息一时难以辨别是否属于以上所列内容之一的，应报相关主管部门审核同意后再发布。

（9） 对客户的个人信息保密，未经客户同意不得向他人泄漏，但法律规定的除外。

本人/本公司保证：在使用本软件进行信息发布活动中，服从监督和管理；若未做到上述一至八条，本人/本公司愿意承担由此引起的一切法律责任，并接受相应的处罚。

目录

目录

版本更新记录

版本更新记录				
更新版本	更新日期	制作人	审核人	更新说明
V1.1	**2016-10-02**	Logan	Frank	初次创建
V1.2	**2017-05-15**	Logan	Frank	添加 https 接口说明
V1.3	**2017-11-09**	Logan	Frank	添加加密接口
V1.4	**2017-12-04**	Logan	Frank	添加扩展号发送接口 添加回复接口 2
V1.5	**2018-05-21**	Logan	Frank	添加 Web Service 接口地址 添加 soap 访问示例
V1.6	**2019-01-28**	Logan	Frank	回执回复返回 customerUuid 字段
V1.7	**2020-06-17**	Logan	Frank	接口描述修改为最新
V1.8	**2020-07-14**	Logan	Frank	新增个性化短信发送接口

1、接口地址

接口开发函数说明

http 服务引用地址为：

http://118.178.116.15

https 服务引用地址为：

https://118.178.116.15:8443

Web Service 服务引用地址为：

http://118.178.116.15/winnerrxd/api/ws/trigger?wsdl

WEB 登录地址为：

http://cloud.winnerlook.com

每个方法均支持 **get**、**post** 接入，请注意 **get** 调用长度控制

接口列表

Command	Function	Method
/winnerrxd/api/trigger/GetBalanceConnect	获取当前账户余额	GET,POST
/winnerrxd/api/trigger/SendMsg	发送短信	GET,POST
/winnerrxd/api/trigger/GetMoConnect	获取短信发送后的回复	GET,POST
/winnerrxd/api/trigger/GetMo2Connect	获取短信发送后的回复	GET,POST
/winnerrxd/api/trigger/GetReportConnect	获取短信发送状态	GET,POST

/winnerrxd/api/trigger/GetReport2Connect	获取短信发送状态	GET,POST
/winnerrxd/api/trigger/SendBatchMsg	批量发送相同内容短信	GET,POST
/winnerrxd/api/trigger/sendMsgByEncrypt	短信加密发送	GET,POST
/winnerrxd/api/trigger/GetReport2ByEncrypt	加密获取短信发送状态	GET,POST
/winnerrxd/api/trigger/GetMo2ByEncrypt	加密获取短信发送后的回复	GET,POST
/winnerrxd/api/trigger/sendMsgExt	短信扩展发送	GET,POST
/winnerrxd/api/trigger/SendIndividualMsg	短信发送(个性化)	GET,POST

2、 接口详细说明

2.1 获取余额*

Command:/winnerrxd/api/trigger/GetBalanceConnect

Method: GET,POST

Description: 获取当前账户的余额

@input

参数名称	类型	描述
userCode	string	登录名称，必填
userPass	string	登录密码，必填

@return

参数名称	类型	描述
	string	成功：返回余额。失败:返回错误码

失败码：参考失败状态码
返回示例： HTTP/1.1 200 OK Content-Type: text/xml; charset=utf-8 Content-Length: length <?xml version="1.0" encoding="utf-8"?> <string xmlns="http://tempuri.org/">100</string>

@前置规则：

验证用户登录账户密码，验证用户绑定 ip。

2.2 短信发送*

Command:/winnerrxd/api/trigger/SendMsg

Method: GET,POST

Description: 短信发送

@input

参数名称	类型	描述
userCode	string	登录名称
userPass	string	登录密码
DesNo	string	手机号码(每次只能提交 1 个号码)
Msg	string	短信内容
autograph	string	签名编号，非必填，留空
customerUuid	string	用户标识字段，非必填（最大长度 60 位）
smsType	Integer	短信类型，非必传。 未指定类型可填写 101

@return

参数名称	类型	描述

	string	成功：返回本次提交的批次号。失败:返回错误码
失败码：参考失败状态码		

返回示例：

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string  xmlns="http://tempuri.org/">22000014763584751 97</string>
```

2.3 获取回复*

Command:/winnerrxd/api/trigger/GetMoConnect

Method: GET,POST

Description: 获取短信发送后的回复

@input

参数名称	类型	描述
userCode	string	登录名称，必填
userPass	string	登录密码，必填

@return

参数名称	类型	描述
	string	返回值：回复信息 **数据格式** ： A 号码|,|A 回复内容|,|A 回复时间|,|A 自定义 ID|;| B 号码|,|B 回复内容|,|B 回复时间|,|B 自定义 ID…… 1) 每个号码及其回复、回复时间称为一组回复，号码、回复内容和回复时间之间用“|,|”分隔；

		2) 每组回复之间，用“|;|”分隔；自定义 ID 为发送时用户的 customerUuid。 3) 没有回复时，返回空字符串。
失败码：参考失败状态码		
返回示例： HTTP/1.1 200 OK Content-Type: text/xml; charset=utf-8 Content-Length: length <?xml version="1.0" encoding="utf-8"?> <string xmlns="http://tempuri.org/">13816025285|,|宋超测试的 MO 信息！123456789|,|2016/10/13 13:51:08|,|yj001</string>		

@前置规则：

验证用户登录账户密码，验证用户绑定 ip。

2.4 获取回复 2*

Command:/winnerrxd/api/trigger/GetMo2Connect

Method: GET, POST

Description: 获取短信发送后的回复（带短信下发号码）

@input

参数名称	类型	描述
userCode	string	登录名称，必填
userPass	string	登录密码，必填

@return

参数名称	类型	描述
	string	返回值：回复信息。失败:返回失败码 **数据格式**： A 号码|,|A 回复内容|,|A 回复时间|,|A 下发号码|,|A 自定义 ID |;| B 号码|,|B 回复内容|,|B

		回复时间|,|B 下发号码|,|B 自定义 ID…… 1) 每个号码及其回复、回复时间、下发号码称为一组回复，号码、回复内容、回复时间和下发号码之间用“|,|”分隔； 2) 每组回复之间，用“|;|”分隔；自定义 ID 为发送时用户的 customerUuid。 3) 没有回复时，返回空字符串。
失败码：参考失败状态码		

返回示例：

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://tempuri.org/">13816025285|,|宋超测试的 MO 信息！12345|,|2016/10/13 13:51:08|,|106911378882931|,|yj001</string>
```

@前置规则：

验证用户登录账户密码，验证用户绑定 ip。

2.5 获取状态报告*

Command:/winnerrxd/api/trigger/GetReportConnect

Method: GET,POST

Description: 获取短信发送后的发送状态

注：只保存 3 天内的状态。 每次最多返回 500 条未获取的状态。

@input

参数名称	类型	描述
userCode	string	登录名称，必填

userPass	string	登录密码，必填
batchNumber	string	备用参数，非必填，留空

@return

参数名称	类型	描述
	string	**数据格式**： 批次 A,号码 A,状态 A, 自定义 ID A| 批次 B, 号码 B,状态 B, 自定义 ID B | 批次 C,号码 C,状态 C, 自定义 ID C | …… 1） 每个号码及其状态称为一组状态，号码和状态间用英文半角逗号(“,”)分隔，自定义 ID 为发送时用户的 customerUuid。 2） 每组状态之间，用“|”分隔； 如： 2314357620085030624,13900000000,DELIVRD,yj001| 2314357620085030653,13900000001,DELIVRD,yj002| 2314357620085030623,13900000002,DELIVRD,yj003| 2314357620085030667,13900000003,DELIVRD,yj004| 2314357620085030665,13900000004,UNDELIVRD,yj005| 2314357620085030629, 13900000005,DELIVRD,yj006| 3） 没有状态时返回空字符串。
失败码：参考失败状态码		

返回示例：

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string  xmlns="http://tempuri.org/">2314357620085030624,
13900000000,DELIVRD,yj001|</string>
```

@前置规则：

验证用户登录账户密码，验证用户绑定 ip。

2.6 获取状态报告 2*

Command:/winnerrxd/api/trigger/GetReport2Connect

Method: GET,POST

Description: 获取短信发送后的发送状态（带回执时间）

注：只保存 3 天内的状态。 每次最多返回 500 条未获取的状态。

@input

参数名称	类型	描述
userCode	string	登录名称，必填
userPass	string	登录密码，必填

@return

参数名称	类型	描述
	string	**数据格式**：批次 A,号码 A,时间 A,状态 A,自定义 ID A| 批次 B,号码 B, 时间 B,状态 B,自定义 ID B | 批次 C,号码 C, 时间 C,状态 C,自定义 ID C | …… 1） 每个号码及其状态称为一组状态，号码和状态间用英文半角逗号(“,”)分隔，自定义 ID 为发送时用户的 customerUuid。 2） 每组状态之间，用“|”分隔； 如： 2314357620085030624,13900000000, 2014/06/10 15:34:11, DELIVRD,yj001| 2314357620085030653,13900000001, 2014/06/10 15:34:11, DELIVRD,yj002| 2314357620085030623,13900000002, 2014/06/10 15:34:11, DELIVRD,yj003| 2314357620085030667,13900000003, 2014/06/10 15:34:11, DELIVRD,yj004| 2314357620085030665, 13900000004, 2014/06/10 15:34:11, UNDELIVRD,yj005| 2314357620085030629, 13900000005, 2014/06/10 15:34:11, DELIVRD,yj006| 3） 没有状态时返回空字符串。
失败码：参考失败状态码		

返回示例：

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string
xmlns="http://tempuri.org/">2314357620085030624,13900000000,
2014/06/10 15:34:11, DELIVRD,yj001|</string>
```

@前置规则：

验证用户登录账户密码，验证用户绑定 ip。

2.7 批量发送相同内容短信

Command:/winnerrxd/api/trigger/SendBatchMsg

Method: GET,POST

Description: 批量发送相同内容短信

@input

参数名称	类型	描述
userCode	string	登录名称，必填
userPass	string	登录密码，必填
DesNo	string	手机号码，必填 多个号码用英文半角逗号分隔。如： 13900000000,13900000001 每次不要超过 50 个号码
Msg	string	短信内容，必填。 (Get 或者 Post 方式注意要用 UTF8 编码)
autograph	string	签名编号，非必填，留空
smsType	Integer	短信类型，非必传。 未指定类型可填写 101

@return

参数名称	类型	描述
	string	成功：返回本次提交的批次号。失败:返回错误码
失败码：参考失败状态码		

返回示例：

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string  xmlns="http://tempuri.org/">22000014763584751 98</string>
```

2.8 短信加密发送

Command:/winnerrxd/api/trigger/sendMsgByEncrypt

Method: GET,POST

Description: 加密发送短信

@input

参数名称	类型	描述
userCode	string	登录名称，必填
submitInfo	string	加密字符串串，详见下面说明，必填
autograph	string	签名编号，非必填，留空
submitInfo 是对以下参数进行组合然后进行 DES 加密后的加密字符串串， 参数进行组合的格式为： userPass=< userPass >&DesNo=< DesNo >&Msg=< Msg >&smsType=< smsType >&Channel=< Channel > DES 加密的 Key 和 IV 均为密码 userPass 的 SHA1 加密（ 40 位大写）的前 8 位 其中的参数说明如下		
userPass	string	登录密码，必填
DesNo	string	手机号码。必填 每次只能提交 1 个号码
Msg	string	短信内容。必填
smsType	Integer	短信类型，非必传。 未指定类型可填写 101
Channel	string	留空或输入 0，兼容用。非必填

示例：

userPass=123&DesNo=18616300000&Msg=你的验证码是:2345【饿了么】&smsType=101&Channel=0

密码 userPass： 123 的 SHA1 码为： 40BD001563085FC35165329EA1FF5C5ECBDBBEEF，前 8 位为 40BD0015，对其进行 DES 加密（ key 和 IV 均为 40BD0015）后，发送时 submitInfo 为：

9ECD998D9F60F7963E137DBC61734970FEE419418FBD2E1622286696AD70D909BF58BB270916073B0464030D4FC52D4AF06BF97B34D7CCB4D06AD49E87576B30CF629652BD44258CDA49DB959914CE4F951879939D6586AE

@return

参数名称	类型	描述
	string	成功：返回本次提交的批次号。失败:返回错误码
失败码：参考失败状态码		

返回示例：

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string  xmlns="http://tempuri.org/">22000014763584751 98</string>
```

@前置规则：

验证用户登录账户密码，验证用户绑定 ip。

2.9 加密获取状态报告

Command:/winnerrxd/api/trigger/GetReport2ByEncrypt

Method: GET,POST

Description: 加密获取短信发送后的发送状态

注：只保存 3 天内的状态。 每次最多返回 500 条未取的状态。

@input

参数名称	类型	描述
userCode	string	登录名称，必填
submitInfo	string	提交的加密字符串，必填
submitInfo 是对以下参数进行组合然后进行 DES 加密后的加密字符串串，		

参数进行组合的格式为： userPass=< userPass > DES 加密的 Key 和 IV 均为密码 userPass 的 SHA1 加密（40 位大写）的前 8 位		
userPass	string	登录密码，必填

@return

参数名称	类型	描述
	string	**数据格式**：批次 A,号码 A,时间 A,状态 A,自定义 ID A| 批次 B,号码 B，时间 B,状态 B,自定义 ID B| 批次 C,号码 C，时间 C,状态 C,自定义 ID C| …… 1） 每个号码及其状态称为一组状态，号码和状态间用英文半角逗号(“,”)分隔，自定义 ID 为发送时用户的 customerUuid。 2） 每组状态之间，用“|”分隔； 如： 2314357620085030624,13900000000,2014/06/10 15:34:11, DELIVRD,yj001| 2314357620085030653,13900000001,2014/06/10 15:34:11, DELIVRD,yj002| 2314357620085030623,13900000002,2014/06/10 15:34:11, DELIVRD,yj003| 2314357620085030667,13900000003,2014/06/10 15:34:11, DELIVRD,yj004| 2314357620085030665, 13900000004, 2014/06/10 15:34:11, UNDELIVRD,yj005| 2314357620085030629, 13900000005, 2014/06/10 15:34:11, DELIVRD,yj006| 3） 没有状态时返回空字符串。
失败码：参考失败状态码		

返回示例：

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string
xmlns="http://tempuri.org/">2314357620085030624,13900000000,
2014/06/10 15:34:11, DELIVRD,yj001|</string>
```

@前置规则：

验证用户登录账户密码，验证用户绑定 ip。

2.10 加密获取回复

Command:/winnerrxd/api/trigger/GetMo2ByEncrypt

Method: GET,POST

Description: 获取短信发送后的回复

@input

参数名称	类型	描述
userCode	string	登录名称，必填
submitInfo	string	提交的加密字符串，必填
submitInfo 是对以下参数进行组合然后进行 DES 加密后的加密字符串串， 参数进行组合的格式为： userPass=< userPass > DES 加密的 Key 和 IV 均为密码 userPass 的 SHA1 加密（40 位大写）的前 8 位		
userPass	string	登录密码，必填

@return

参数名称	类型	描述
	string	返回值：回复信息 **数据格式** ： A 号码|,|A 回复内容|,|A 回复时间|,|A 自定义 ID|;| B 号码|,|B 回复内容|,|B 回复时间|,|B 自定义 ID……

		1） 每个号码及其回复、回复时间称为一组回复，号码、回复内容和回复时间之间用“|,|”分隔； 2） 每组回复之间，用“|;|”分隔；自定义 ID 为发送时用户的 customerUuid。 3） 没有回复时，返回空字符串。
失败码：参考失败状态码		

返回示例：

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string  xmlns="http://tempuri.org/">13816025285|,|宋超测试的 MO 信息！ 123456789|,|2016/10/13 13:51:08|,|yj001</string>
```

@前置规则：

验证用户登录账户密码，验证用户绑定 ip。

2.11 短信扩展发送*

Command: /winnerrxd/api/trigger/sendMsgExt

Method: GET,POST

Description:带自定义扩展号短信发送

@input

参数名称	类型	描述
userCode	string	登录名称，必填
userPass	string	登录密码，必填
DesNo	string	手机号码(每次只能提交 1 个号码)
Msg	string	短信内容，必填。

		(Get 或者 Post 方式注意要用 UTF8 编码)
autograph	string	签名编号，非必填，留空
ExeNo	string	用户自定义扩展号(1-16 位数字格式),可为空，非必填
customerUuid	string	用户标识字段，非必填（最大长度 60 位)
smsType	Integer	短信类型，非必传。 未指定类型可填写 101

@return

参数名称	类型	描述
	string	成功：返回本次提交的批次号。失败:返回错误码
失败码：参考失败状态码		

返回示例：

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string  xmlns="http://tempuri.org/">220001476358475197</string>
```

@前置规则：

验证用户登录账户密码，验证用户绑定 ip。

2.12 短信发送(个性化)*

Command: /winnerrxd/api/trigger/SendIndividualMsg

Method: GET ,POST

Description: 短信发送(个性化)

@input

参数名称	类型	描述
userCode	string	登录名称，必填

userPass	string	登录密码，必填
Msg	string	短信内容，必填（短信内容为 utf-8 编码,每条个性化短信以英文 |-| 分割 ,号码与短信内容以 英文|~| 分割） 每次不要超过 50 条短信
smsType	Integer	短信类型，非必传。未指定类型可填写 101
customerUuid	string	用户标识字段，非必填（最大长度 60 位）

@return

参数名称	类型	描述
	string	成功：返回本次提交的批次号。失败:返回失败码
失败码：参考失败状态码		

返回示例：

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string  xmlns="http://tempuri.org/">22000014763584751 97</string>
```

@前置规则：

验证用户登录账户密码，验证用户绑定 ip。

3、失败状态码

失败状态码对应说明：

失败状态码	状态码说明

-1	应用程序错误
-3	用户名或密码错误
-4	短信内容和备案的模板不一样
-5	签名不正确
-7	余额不足
-8	无可用通道或不在通道时间范围
-9	无效号码
-10	签名内容不符合长度
-11	用户有效期过期
-12	黑名单
-16	接口请求过于频繁，余额接口 5s 秒一次，其他接口适当调整
-17	非法 IP
-18	Msg 内容格式错误
-19	短信数量错误,小于 1 /大于 500(个性化)/大于 1000（群发）
-20	号码错误或者黑名单
-23	解密失败
-24	短信包含用户敏感信息
-25	用户被冻结
-26	无效数据
-27	请求参数错误
-28	无效数据
-41	指定短信模板类型错误或短信类型参数错误
-44	自定义扩展号不符合规则（1-16 位数字）
-46	用户黑名单
-47	系统黑名单
-48	号码超频拦截
-51	超过设置的每月短信条数的限制
-54	短信包含系统敏感信息

4、请求示例

以下 Java 代码为短信发送接口请求示例，执行后会打印出请求结果，如需测试其它接口，需替换请求地址(url)和请求参数变量

说明：此处需引用 httpclient、httpcore、commons-logging 三个 jar 包

http 示例代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.*;
import java.security.MessageDigest;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.*;
import javax.crypto.SecretKey;
import javax.crypto.spec.DESKeySpec;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.SecretKeyFactory;
import javax.crypto.Cipher;

public static void main(String[] args) {
    String url="http://118.178.116.15/winnerrxd/api/trigger/SendMsg";


    List<NameValuePair> nvps = new ArrayList<NameValuePair>();
    nvps.add(new BasicNameValuePair("userCode", "用户名"));
    nvps.add(new BasicNameValuePair("userPass", "密码"));
```

```
            nvps.add(new BasicNameValuePair("DesNo", "手机号"));
            nvps.add(new BasicNameValuePair("Msg", "短信内容【签名】"));
            String post=httpPost(url,nvps);  //post 请求

            String getparam="userCode=URLEncoder.encode(用户名)&userPass=URLEncode
r.encode(密码)&DesNo=URLEncoder.encode(手机号)&Msg=URLEncoder.encode(短信内容【签
名】) ";
            String result=httpGet(url,getparam); //get 请求
        }

    public static String httpPost(String url,List<NameValuePair> params) {
            String result = "";
        try {
            HttpClient httpclient = new DefaultHttpClient();
            HttpPost httpPost = new HttpPost(url);
            httpPost.setEntity(new UrlEncodedFormEntity(params, "UTF-8"));
            HttpResponse response = httpclient.execute(httpPost);
            HttpEntity entity = response.getEntity();
            if (entity != null) {
                 InputStream instreams = entity.getContent();
                 result = convertStreamToString(instreams);
                 System.out.println(result);
             }
        } catch (Exception e) {
                e.printStackTrace();
        }
        return result;
    }

    public static String httpGet(String url,String params){
        String result="";
        try{
            HttpClient client=new DefaultHttpClient();
            if(params!=""){
                url=url+"?"+params;
            }
            HttpGet httpget=new HttpGet(url);
            HttpResponse response=client.execute(httpget);
            HttpEntity entity=response.getEntity();
            if (entity != null) {
                 InputStream instreams = entity.getContent();
                 result = convertStreamToString(instreams);
                 System.out.println(result);
```

```
            }
        }catch(Exception e){
                e.printStackTrace();
        }
        return result;
    }

    public static String convertStreamToString(InputStream is) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(is));

        StringBuilder sb = new StringBuilder();

        String line = null;
        try {
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                is.close();
            } catch (IOException e) {
               e.printStackTrace();
            }
        }
        return sb.toString();
    }
```

https 示例代码：

```
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.cert.CertificateException;
import java.util.HashMap;
import java.util.Map;
import javax.net.ssl.HostnameVerifier;
```

```
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import java.security.cert.X509Certificate;

public static void main(String[] args) {
        String url = "https://118.178.116.15:8443/winnerrxd/api/trigger/SendMsg ";
        Map<String, String> paramMap = new HashMap<String, String>();
        paramMap.put("userCode", "用户名");
        paramMap.put("userPass", "密码");
        paramMap.put("DesNo", "手机号");
        paramMap.put("Msg", "短信内容【签名】");
        paramMap.put("smsType","短信类型")

        try {
            //GET请求
            httpsGet(url, paramMap);
            //POST请求
            httpsPost(url, paramMap);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.err.println("调用https失败:"+e);
        }
       }
private static class TrustAnyTrustManager implements X509TrustManager {

        public void checkClientTrusted(X509Certificate[] chain, String authType)
                throws CertificateException {
        }

        public void checkServerTrusted(X509Certificate[] chain, String authType)
                throws CertificateException {
        }

        public X509Certificate[] getAcceptedIssuers() {
            return new X509Certificate[] {};
        }
    }

    private static class TrustAnyHostnameVerifier implements HostnameVerifier {
```

```
        public boolean verify(String hostname, SSLSession session) {
            return true;
        }
    }

    public static String httpsGet(String url,Map<String, String> params) throws
Exception
    {
         String result = "";
         BufferedReader in = null;
            try {

                String urlStr = url + "?" + getParamStr(params);

                System.out.println("GET请求的URL为: "+urlStr);
                SSLContext sc = SSLContext.getInstance("SSL");
                sc.init(null, new TrustManager[] { new TrustAnyTrustManager() },
                        new java.security.SecureRandom());
                URL realUrl = new URL(urlStr);
                // 打开和URL之间的连接
                HttpsURLConnection connection = (HttpsURLConnection)
realUrl.openConnection();
                //设置https相关属性
                connection.setSSLSocketFactory(sc.getSocketFactory());
                connection.setHostnameVerifier(new TrustAnyHostnameVerifier());
                connection.setDoOutput(true);

                // 设置通用的请求属性
                connection.setRequestProperty("accept", "*/*");
                connection.setRequestProperty("connection", "Keep-Alive");
                connection.setRequestProperty("user-agent",
                        "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;SV1)");
                // 建立实际的连接
                connection.connect();

                // 定义 BufferedReader输入流来读取URL的响应
                in = new BufferedReader(new
InputStreamReader(connection.getInputStream(),"UTF-8"));
                String line;
                while ((line = in.readLine()) != null) {
                    result += line;
                }
```

```
System.out.println("获取的结果为：“+result);
            } catch (Exception e) {
                System.out.println("发送GET请求出现异常！" + e);
                //e.printStackTrace();
                throw e;
            }
            // 使用finally块来关闭输入流
            finally {
                try {
                    if (in != null) {
                        in.close();
                    }
                } catch (Exception e2) {
                    //e2.printStackTrace();
                    throw e2;
                }
            }
            return result;

    }

    public static String httpsPost(String url, Map<String, String> params)
            throws Exception{
         String result = "";
          BufferedReader in = null;
          String content = getParamStr(params);
          try {
           System.out.println("POST请求的URL为：“+url);
            SSLContext sc = SSLContext.getInstance("SSL");
            sc.init(null, new TrustManager[] { new TrustAnyTrustManager() },
                    new java.security.SecureRandom());
            URL console = new URL(url);
            HttpsURLConnection connection = (HttpsURLConnection)
console.openConnection();
            connection.setSSLSocketFactory(sc.getSocketFactory());
            connection.setHostnameVerifier(new TrustAnyHostnameVerifier());
            connection.setDoOutput(true);
            connection.connect();
            DataOutputStream out = new
DataOutputStream(connection.getOutputStream());
            out.write(content.getBytes("UTF-8"));
            // 刷新、关闭
            out.flush();
```

```
            out.close();
            // 定义 BufferedReader输入流来读取URL的响应
            in = new BufferedReader(new
InputStreamReader(connection.getInputStream(),"UTF-8"));
            String line;
            while ((line = in.readLine()) != null) {
                result += line;
            }
            System.out.println("获取的结果为："+result);
          } catch (Exception e) {
            System.out.println("发送POST请求出现异常！" + e);
              //e.printStackTrace();
              throw e;
       }
          // 使用finally块来关闭输入流
          finally {
              try {
                  if (in != null) {
                      in.close();
                  }
              } catch (Exception e2) {
                  //e2.printStackTrace();
                  throw e2;
              }
          }
        return result;
    }

    private static String getParamStr(Map<String, String> params)
    {
       String paramStr="";
       if(params != null && params.size() >0){
          // 获取参数列表组成参数字符串
          for (String key : params.keySet()) {
              paramStr+=key+"="+params.get(key)+"&";
          }
          //去除最后一个"&"
          paramStr=paramStr.substring(0, paramStr.length()-1);
       }
        return paramStr;
    }
```

SOAP 请求示例代码：

以短信发送 soap 为例,参数填写参考接口的具体参数类型

SOAP 1.1

以下是 SOAP 1.1 请求和响应示例。所显示的占位符需替换为实际值。

```
POST /winnerrxd/api/ws/trigger HTTP/1.1
Host: 118.178.116.15
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: http://tempuri.org/SendMsg

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SendMsg xmlns="http://tempuri.org/">
      <userCode>string</userCode>
      <userPass>string</userPass>
      <DesNo>string</DesNo>
      <Msg>string</Msg>
      <autograph>string</autograph>
    </SendMsg>
  </soap:Body>
</soap:Envelope>


HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SendMsgResponse xmlns="http://tempuri.org/">
      <SendMsgResult>string</SendMsgResult>
    </SendMsgResponse>
```

```
  </soap:Body>
</soap:Envelope>
```

SOAP 1.2

以下是 SOAP 1.2 请求和响应示例。所显示的占位符需替换为实际值。

```
POST /winnerrxd/api/ws/trigger HTTP/1.1
Host: 118.178.116.15
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <SendMsg xmlns="http://tempuri.org/">
      <userCode>string</userCode>
      <userPass>string</userPass>
      <DesNo>string</DesNo>
      <Msg>string</Msg>
      <autograph>string</autograph>
    </SendMsg>
  </soap12:Body>
</soap12:Envelope>


HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <SendMsgResponse xmlns="http://tempuri.org/">
      <SendMsgResult>string</SendMsgResult>
    </SendMsgResponse>
  </soap12:Body>
</soap12:Envelope>
```

5、回执推送

5.1 回执主动推送格式

- Method: POST
- Description: 用户需部署接收服务器，在设置触发回执的接收 API 接口地址后，即可开启，数据传输格式为 Json。（请联系客服配置根用户推送接口地址）
- **@Input**

参数名称	类型	描述
json_str	@RequestBody string	回执回复 json 字符串

- **@Return**

参数名称	类型	描述
	string	返回接收结果

5.2 回执主动推送 json_str 说明

返回内容			JSON 文本			
一级	**二级**	**接口字段名**	**字段编码**	**字段类型**	**字段长度**	**取值描述**
{}		用户名	account	string		
		回执类型	type	Integer		1:触发回执 2:群发回执
	list[]	回执列表	list	list		
		手机号	phone	string		
		回执状态	receive_status	string		
		接收回执时间	receive_time	Date		

		批次号	sub_no	string		
		用户标识	customer_uuid	string		

回执 json_str 序列化对象格式示例：


```
{
 "list": [{
  "phone": "15021750000",
  "receive_status": "DELIVRD",
  "receive_time": 1529466210680,
  "sub_no": "2400350000122303487",
  "customer_uuid":"0000a8ae8c1544e8b5723d77882d1280"
 }],
 "account": "zhangsan",
 "type": 1
}
```


6、回复推送

6.1 回复主动推送格式

- Method: POST
- Description: 用户需部署接收服务器，在设置触发回复的接收 API 接口地址后，即可开启，数据传输格式为 Json。（请联系客服配置根用户推送接口地址）
- **@Input**

参数名称	类型	描述
json_str	@RequestBody string	回执回复 json 字符串

- **@Return**

参数名称	类型	描述
	string	返回接收结果

6.2 回复主动推送 json_str 说明

返回内容			JSON 文本			
一级	二级	接口字段名	字段编码	字段类型	字段长度	取值描述
{}		用户名	account	string		
		回执类型	type	Integer		3:触发回复 4:群发回复
	list[]	回执数据	list	list		
		回复号码	dest_phone	string		
		下发号码	show_num	string		
		回复内容	resp_content	string		
		回复时间	resp_time	Date		
		用户标识	customer_uuid	string		

回复 json_str 序列化对象格式示例：

```
{
 "list": [{
  "dest_phone": "15021750000",
  "resp_content": "回复内容",
  "receive_time": 1529466210680,
  "show_num": "10690835400387",
  "customer_uuid":"0000a8ae8c1544e8b5723d77882d1280"
 }],
 "account": "zhangsan",
 "type": 3
}
```