

凌凯直连接口开发说明

(V17.11)

1. HTTP 协议

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

参数	类型	说明	参考
dc	integer(1)	0 表示英文, 8 表示 UCS2, 15 表示中文	3.1
ec	integer(1)	一般不用	3.2
sm	string	默认 HEX 编码之消息内容; 客户可以指定形式	3.3
pi	integer(1)	一般不用	3.4
da	string	手机号	3.5
sa	string	扩展码, 必须以账号设定的开头; 可以不填写	3.6
ld	string	linkID, 目前不用	3.7
ex	integer(8)	外部编码, 长整型	3.8
rd	integer(1)	是否需要状态报告	3.9
un	string	用户名	3.10
pw	string	密码, 切勿直接使用, 注意安全	3.11
st	string	定时发送时间; 可不填	3.12
mu	string (32)	模块名, 一般不用	3.13
pr	integer(1)	优先级	3.14
vp	string	有效期, 可不填	3.15
rf	integer(1)	控制返回格式	3.16
ts	string	时间标记, 用于验证, 格式 <code>yyyyMMddHHmmss</code>	3.26
tf	integer(1)	短信内容的传输编码, 默认为 0 表示 HEX 格式	3.28
rl	integer(4)		

颜色标出为必须填写发送示例, 消息内容为 “ABC” (414243 为十六进制表示), 需要状态报告(`rd=1`):

<http://IP+Port /mt?un=star&pw=123456&da=13612345678&sm=414243&dc=15&rd=1> 或者这样发送, 内容为 “ABC”, 设置 `tf=2`; 需使用 URLEncoder:

<http://IP+Port /mt?un=star&pw=123456&da=13612345678&sm=ABC&dc=15&tf=2>

不同号码不同内容群发

da	string	空（或不存在），此时默认分割符“ ”（半角）；或者分隔符字符串，长度 1~4， 必须半角 ，比如“\$\$”，但不可以为“#”	3.5
sm	string	默认 HEX 编码之消息内容；编码前格式为“手机号#外部编号#内容 手机号#外部编号#内容”，其中的分割符“ ”可以通过参数 da 来指定。分割符后面不可以有空格。如果没有指明外部编号的，则使用参数“ex”的设置，如果存在的话。“外部编号”定义同参数“ex”定义。 举例：“13812345678#1234#测试短信” 详细请查看实例代码。	3.3

使用该方式发送时，请务必确保手机号正确，内容合法。

发送结果 , 字段含义如下:

参数	类型	说明	参考
r	integer(4)	错误码	3.17
id	string	消息编号，成功时返回，失败时该字段可省略	3.18

发送结果返回举例:

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

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

接收短信入口 <http://<host>:<port>/mo>

接收状态报告和上行信息都在该地址

参数	类型	说明	参考
un	string	用户名	3.10
pw	string	密码	3.11

fs	integer(4)	返回大小设定	3.27
ts	string	时间标记, 用于验证, 格式 yyyyMMddHHmmss	3.26
rf	integer(1)	控制返回格式	3.16
tf	integer(10)	短信内容的传输编码, 默认为 0 表示 HEX 格式	3.28

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

行间参数用&分割

参数	类型	上行消息说明	状态报告说明	参考
op	string	mo	dr	3.19
dc	integer(1)	消息类型		3.1
pi	integer(1)	cmpp 之 tp_pid		3.4
ec	integer(1)	cmpp 之 tp_udhi		3.2
sa	string	手机号		3.6
da	string	扩展码		3.5
mu	string	模块名		3.13
sm	string	HEX 编码之消息内容		3.3
id	string		消息编号	3.18
ex	integer(8)		外部编码	3.8
su	string		状态说明	3.20
sd	string	接收时间	提交时间	3.21
dd	string		完成时间	3.22
rp	integer(4)		错误码	3.23
bi	integer(4)		拆分编号	3.24
di	integer(4)		群发时号码的位置	3.25

除 op 之外, 状态报告和 mo 的各个参数未必都存在, 特别是蓝色标出的。

客户也可根据需要申请主动推送之 **http** 接口, 即由我方服务器向客户方服务器主动发送 **http** 请求, 每次一个数据; 参数如上说明, 不再赘述。这是推荐方式, 可以保证上行和状态报告的及时性。

关于群发时装态报告返回的编号 id, 一般为提交时返回的编号, 客户可以以该 id 和手机号为规则进行对应; 表中<bi><di>分别和内容拆分和号码拆分有关, 如果存在都是从 1 开始。

如果账号设置了需要连续编号, 则对于群发 n 个号码, 状态报告返回的编号为 id+0 到 id+n-1。不建议群发。

接收上行和状态报告举例

<http://IP+Port/mo?un=star&pw=123456> 如需

返回 xml 格式的响应

<http://IP+Port/mt?un=star&pw=123456&rf=1>

查询余额举例:

<http://IP+Port/bi?un=star&pw=123456>

举例 1: 加密未经处理, 使用 HEX 发送式 (java+httpClient4+codec):

```
String mobile = "13612345678"; // 手机号
String username = "test"; // 用户名

String password = "123456"; // 密码      int
dataCoding = 8; // UNICODE 编码 (UTF-16BE)
// 使用 Hex 编码内容

String message = "这是一条测试短信";
String hex = Hex.encodeHexString( message.getBytes("UTF-16BE") );
StringBuilder sb = new StringBuilder(); sb.append("http://localhost:7891/mt?")
    .append("un=").append( username )
    .append("&pw=").append( password )
    .append("&da=").append( mobile )
    .append("&dc=").append( dataCoding )
    .append("&sm=").append( hex );

String req = sb.toString();
System.out.println("request: " + req);

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

发送成功的应答如下:

```
id=142551080979073768
```

举例 2: 密码经过 MD5 处理, 使用 URLEncode+UTF8 (java+httpClient4):

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
String timestamp = sdf.format(new Date() );
String mobile = "13612345678"; // 手机号
String shortCode = "8888"; // 扩展码
String username = "test"; // 用户名
```

```
String password = "123456"; // 密码

long externalId = 0x123456789L; // 自定义消息编码, 可以忽略

Int dataCoding = 8; // UNICODE 编码

int transferEncoding = 3; // URLEncode+UTF8

int responseFormat = 2; // 返回格式为 Json 格式

String message = "这是一条测试短信,返回 Json";

// 计算密码摘要

SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");

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

MessageDigest md5 = MessageDigest.getInstance("MD5");

md5.update(username.getBytes("utf8"));

md5.update(password.getBytes("utf8"));

md5.update(timestamp.getBytes("utf8"));

md5.update( message.getBytes("utf8") );

password = Base64.encodeBase64String( md5.digest() );

StringBuilder sb = new StringBuilder();

sb.append("http://localhost:7890/mt?")

    .append("un=").append( username )

    .append("&pw=").append( password )

    .append("&ts=").append( timestamp )

    .append("&da=").append( mobile )

    .append("&sa=").append( shortCode )

    .append("&ex=").append( externalId )

    .append("&dc=").append( dataCoding )

    .append("&tf=").append( transferEncoding )

    .append("&rf=").append( responseFormat )

    .append("&sm=").append( URLEncoder.encode(message, "utf8") );

String req = sb.toString();

System.out.println("request: " + req);

String result = Request.Get( req ).connectTimeout(60000).socketTimeout(60000)

    .execute().returnContent().asString();

System.out.println( "response: " + result );
```

返回格式参考 (Json 格式) : {"success": true, "id": "142540128277943843"}

举例 3 : 密码为明码,使用 URLEncode+UTF8 (java+HttpClient4) :

```
String mobile = "13612345678"; // 手机号
String shortCode = "8888"; // 扩展码
String username = "test"; // 用户名
String password = "123456"; // 密码
long externalId = 0x123456789L; // 自定义消息编码, 可以忽略
int dataCoding = 8; // UNICODE 编码
int transferEncoding = 3; //URLEncode+UTF8
int responseFormat = 1; // 返回格式为 xml 格式
String message = "这是一条测试短信, 返回 XML";
StringBuilder sb = new StringBuilder();
sb.append("http://localhost:7891/mt?")
    .append("un=").append( username )
    .append("&pw=").append( password )
    .append("&da=").append( mobile )
    .append("&sa=").append( shortCode )
    .append("&ex=").append( externalId )
    .append("&dc=").append( dataCoding )
    .append("&tf=").append( transferEncoding )
    .append("&rf=").append( responseFormat )
    .append("&sm=").append( URLEncoder.encode(message, "utf8") );

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

返回格式参考 (成功的 XML):

```
<?xml version="1.0"?>
<id>142540261421930021</id>
```

举例 4 : 获得发送结果, 密码为明码:

```
String username = "test"; // 用户名

String password = "123456"; // 密码

int fetchSize = 100; // 每次返回的最大条数

StringBuilder sb = new StringBuilder();

sb.append("http://localhost:7891/mo?")

.append("un=").append( username )

.append("&pw=").append( password )

.append("&fs=").append( fetchSize )

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


String req = sb.toString();

System.out.println("request: " + req);

String result =Request.Get( req ).connectTimeout(60000).socketTimeout(60000)

.execute().returnContent().asString();

System.out.println( "response: " + result );
```

举例 5: 获得发送结果，密码经过 MD5 处理:

```
String username = "test"; // 用户名

String password = "123456"; // 密码

int fetchSize = 100; // 每次返回的最大条数

SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");

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

MessageDigest md5 = MessageDigest.getInstance("MD5");

md5.update( username.getBytes("utf8") );

md5.update(password.getBytes("utf8") );

md5.update( timestamp.getBytes("utf8") );

password = Base64.encodeBase64String( md5.digest() );

StringBuilder sb = new StringBuilder();

sb.append("http://localhost:7890/mo?")

.append("un=").append( username )

.append("&pw=").append( password )

.append("&fs=").append( fetchSize )

.append("&ts=").append( timestamp )

.append("&rf=").append( 1 ); // 控制返回格式，可不加
```

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

举例 6: C#发送:

```
string username = "test";
string password = "123456";
string message = "短信发送测试【测试】";
Encoding enc = Encoding.GetEncoding("UTF-16BE");
message = BitConverter.ToString(enc.GetBytes(message)).Replace("-", "");
string destAddr = "13600000000";
String url = "http://192.168.1.100:7891/mt?un=" + username
    + "&pw=" + password + "&da=" + destAddr + "&dc=8&sm=" + message;
WebClient client = new WebClient();
Stream stream = client.OpenRead(url);
StreamReader r = new StreamReader(stream);
string result = r.ReadToEnd();
stream.Close();
Console.WriteLine(result);
```

举例 7: 中文不同内容短信群发的参数处理 (java):

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

举例 8: UCS2 不同内容短信群发的参数处理 (java):


```
String mobile = "$$"; // 参数 da: 自定义分割符$$
int dataCoding = 8; // 参数 dc: 使用 UCS2 编码, 支持国际语言
int transferEncoding = 2; // 参数 tf: URLEncode

String [] messages = {"第一条 Тестовое сообщение",
                      "第二条 Тестовое сообщение2"};

String[] mobiles = { "1381234001","1381234002" };

StringBuilder msg = new StringBuilder();
for(int i = 0; i < messages.length; i++) {
    if ( i > 0 ) msg.append( mobile );
    msg.append(mobiles[i] ).append("#");
    msg.append( new String(messages[i].getBytes("UTF-16BE"), "ISO8859_1") );
}

String message = msg.toString();

....

.append("&sm=").append( URLEncoder.encode(message, "ISO8859_1") );

....
```

2. 错误码定义

以下为系统定义的错误码列表

错误码	说明
9002	未知命令
9012	短信消息内容错误
9013	目标地址错误
9014	短信内容太长
9015	路由错误
9016	没有下发网关
9017	定时时间错误
9018	有效时间错误
9019	无法拆分或者拆分错误
9020	号码段错误
9021	消息编号错误, 这个和 PacketIndex 参数有关
9022	用户不能发长短信(EsmClass 错误)
9023	ProtocolID 错误
9024	结构错误, 一般是指长短信
9025	短信编码错误
9026	内容不是长短信
9027	签名不对
9028	目标网关不支持长短信

9029	路由拦截
9030	目标地址(手机号)太多
9031	目标地址(手机号)太少
9032	发送速度太快
9101	验证失败，一般和用户名/密码/IP 地址相关
9102	没有填写用户名
9103	名字没找到
9104	IP 地址不对
9105	超过最大连接数，就是 tcp 连接数，http 也是一样的
9106	协议版本错误
9107	帐号无效，比如过期/禁用
9108	功能尚未开通
9902	网关无此能力
9903	二进制数据太长了；如网关没有特别说明，一般不能超过 140，
9904	网关不支持 EsmClass 字段，或等同字段
9905	网关不支持 ProtocolID 字段，或等同字段
9906	网关不支持 UDHI 字段，或等同字段
9907	网关支持 Letter 字段发送，但短信记录没有 letter
9908	网关不存在
9909	网关没有应答
9910	网关不支持该短信编码
9911	区域错误
9401	计费错误
9402	非法内容
9403	黑名单
9404	丢弃
9405	Api 帐号丢失
9406	配置拒绝，就是帐号设置了拒绝标记
9407	帐号没有生成时间,这个属于非法帐号
9408	消息超时，超过短信或帐号或系统设置的生存时间
9409	由约束规则拒绝
9410	状态报告超时
9411	状态报告
9412	帐号无效
9413	重发拦截
9414	转发时丢弃，比如该通道已经废弃
9415	人工审核失败
9416	可能是诈骗信息
9417	不匹配模板
9418	拒绝审核（审核功能可能关闭）
9419	超过该手机号码的日发送次数限制
9420	夜间不可以群发
9421	夜间发送禁止

9422	会审失败
9423	高危短信
9424	非法字符，繁体字等
9425	多签名
9426	扩展码签名错误，必须在签码表内
9427	保留
9428	短信内容只有签名
9429	签名超过长度限制, 即过长或者过短，或者没有签名；注意可能有不可见字符
9430	长短信条数不完整
9431	签名被禁止
9432	超过每分钟最大下发条数
9433	超过每小时最大下发条数
9434	超过全天最大下发次数限制，这个错误码和 9419 是两个不同的限制
9435	超过每日下发极限
9436	网关拒绝发送长短信
9437	签名位置不正确
9438	丢失
9439	扩展失败
9440	钓鱼
9441	长频次
9442	发送条数超过每日限额
9443	长短信未能完成组装；一般是长短信不完整，或者之间时间间隔太长
9444	长短信未能完成组装 2
9501	非法目标地址，即手机号
9502	消息无法投入队列
9503	保留
9504	保留
9601	上行路由失败
9602	超过最大重试
9603	上行处理错误
9701	通知失败
9702	无法创建句柄，一般是个配置错误
9703	消息已经送出，但应答时超时
9801	投递地址错
9802	无法连接到服务器
9803	投递发送数据失败
9804	投递接收结果失败
9805	验证链路/登录失败
9806	接收时关闭
9807	接收时无数据
9808	接收数据格式错误，比如不是 HTTP 数据

仅供参考

3. 名词解释

3.1. data_coding

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

3.2. esm_class

smpp 之 esm_class, 8bits; 一般不用

3.3. content

HEX 编码之消息内容; 例如内容'ABC'经 HEX 后为'414243'; 也可以指定编码格式

3.4. protocol_id

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

3.5. dest_addr

在发送时为手机号码: 多个号码用分号(半角)分割。请不要超过 100 个

在接收上行信息时为扩展码, 一般为数字字符串。如果是不同手机号不同内容群发, 则该参数为空, 或者客户自定义分割符。可参考 http 接口说明。

3.6. source_addr

在发送时为扩展码, 必须以账号设定的开头; 可以不填写。

在接收上行信息时为手机号码

3.7. link_id

目前不用

3.8. external_id

外部编码,长整型; 客户可以自行填写, 状态报告时返回

3.9. registered_delivery

是否需要状态报告; 0 表示不需要; 1 表示需要

3.10 username

用户名

3.11 password

密码。请不要在通讯中直接使用密码, 如必须使用, 请务必提供 **IP** 地址进行锁定。

3.12 schedule_time

定时发送时间; 格式 yyyyMMddHHmmss; 可不填

3.13. module

模块名, 一般不用填写

3.14. priority

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

3.15. validity_period

有效期；格式 yyyyMMddHHmmss；可不填

3.16. response_format

控制返回格式：

0：默认格式。

1：xml，xml 格式中的元素名同 HTTP 接口命名。

2：json 格式

3.17. result

错误码，0 表示成功,其他错误代码；在 0 时该字段可省略

3.18. message_id

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

3.19. operation

取值 mo 或者 dr 表明这是一个上行短信或者状态报告

3.20. status

状态报告描述字符串

3.21. submit_date

消息提交时间

3.22. done_date

消息完成时间

3.23. receipt

状态报告错误码，0 表示用户接收成功。

3.24. block_index

消息发生拆分，该值表明这是第几段消息

3.25. dest_index

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

3.26. timestamp

如果该字段存在，服务器将认为 password 由如下方式计算得出：

`password=Base64(MD5(username+password+timestamp+content))`

其中 username, password, timestamp 以 utf-8 处理成字节数组。拼接时是按字节数组拼接 (content 是按原始短信字节数组处理，并非编码后的字符串)；MD5 之后得到 16 字节 (非长度为 32 的字符串)，再使用 Base64 编码。如果 content 不存在，则不参与计算。

该字段格式 yyyyMMddHHmmss

java 的示例代码如下：

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

MessageDigest md5 = MessageDigest.getInstance("MD5");

md5.update(username.getBytes("utf8"));

md5.update(password.getBytes("utf8"));

md5.update( timestamp.getBytes("utf8") );

if (message != null)
md5.update( message.getBytes("utf8") );

password = Base64.encodeBase64String( md5.digest() );
```

3.27. fetch_size

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

3.28. transfer_encoding

短信内容的传输编码

0: HEX 编码格式(这是默认编码)

1: Base64 编码格式

2: URLEncode 编码（即针对字节进行 URLEncode 编码）

3: URLEncode+UTF8(即原始文本用 UTF8 转字节后进行 URLEncode 编码)举例如下：

	tf 取值	英文消息：“abc”	中文消息：“您好”
HEX	0 或者不设置	616263	c4fabac3
Base64	1	YWJj	xPq6ww==
URLEncode	2	abc	%C4%FA%BA%C3
URLEncode+UTF8	3	abc	%E6%82%A8%E5%A5%BD

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

4. 备注

标准协议均可使用各运营商提供的协议开发包进行对接。

5. 验证码防盗刷注意事项

5.1. 手机号码限制

在程序规定的时间范围内，同一个手机号只能发送程序限定次数的短信；

如若超出程序设定限制，则拒绝此号码发送短信，并且开发人员应根据程序的实际情况做相应处理。

5.2. IP 限制

在程序中获取用户发送短信时的 IP，并作记录，对此 IP 在当天发送短信的次数应作相应限制，以及在规定时间范围内发送短信的频率；

当此 IP 超出调用发送短信次数及时间频率的时候，则拒绝用户通过此 IP 发送短信，并且开发人员应根据程序的实际情况做相应处理。

5.3. 程序客户端调用发送短信后台方法时间间隔限制

当用户在客户端点击“发送短信”按钮发送短信后，应在客户端禁用“发送短信”按钮，并作 30 秒至 1 分钟（具体时间依据自身程序情况决定）计时处理；

在计时未到的时间内，“发送短信”按钮应处于禁用状态，当计时结束才可重新点击“发送短信”按钮。

5.4. 流程限制

用户在不满足程序所规定的流程条件情况下，不允许触发发送短信；举例（用户注册）：

1. 用户注册的用户名需要通过程序自身表单验证后，并返回正确才能进入下一流程；
2. 用户注册设置的密码需要填写，且通过程序表单验证后返回正确，才能进入下一流程；
3. 用户注册输入的手机号码需要通过程序自身的表单验证后，并返回正确才能进入下一流程
4. 用户注册需要验证随机数，用户输入程序产生的随机数，并且两者相匹配，才能进入下一流程；

当用户注册同时满足流程 1、2、3、4 则可以调用发送短信，不满足则禁止发送；注：此点需要根据自己情况，对程序设定流程。

5.5. 图形验证码限制

在用户调用发送短信之前，程序产生图形验证码，需要用户手动输入图形验证码的内容，并且图形验证码验证通过，才能调用发送短信；验证未通过则禁止发送，重新校验图形验证码。

5.6. 注意事项

1. 调用接口发送短信的程序代码，不要暴露在客户端；
2. 将程序所处服务器的 IP 和调用短信接口的账号，交由凌凯客服进行 IP 绑定，这样就限制了接口账号，只能从这个 IP 向接口提交短信。

6. 温馨提示

为保障短信测试的速度性和准确性，请提交测试**短信模版**给业务同事进行绑定测试。