

云通讯平台 PHP 开发指南

- 1 概述
 - 1.1 介绍
 - 1.2 开发总体流程
 - 1.3 安全机制
 - 1.4 平台术语
 - 1.5 参考文档
- 2 快速体验
 - 2.1 申请测试账号
 - 2.2 环境搭建
 - 2.2.1 下载
 - 2.2.2 安装与设置
 - 2.3 Demo介绍
 - 2.4 配置帐号信息
 - 2.5 运行体验
 - 2.5.1 创建子账户
 - 2.5.2 发送短信
 - 2.5.3 双向回呼
- 3 错误码
 - 3.1 云通讯平台错误码
 - 3.2 HTTPS标准错误码

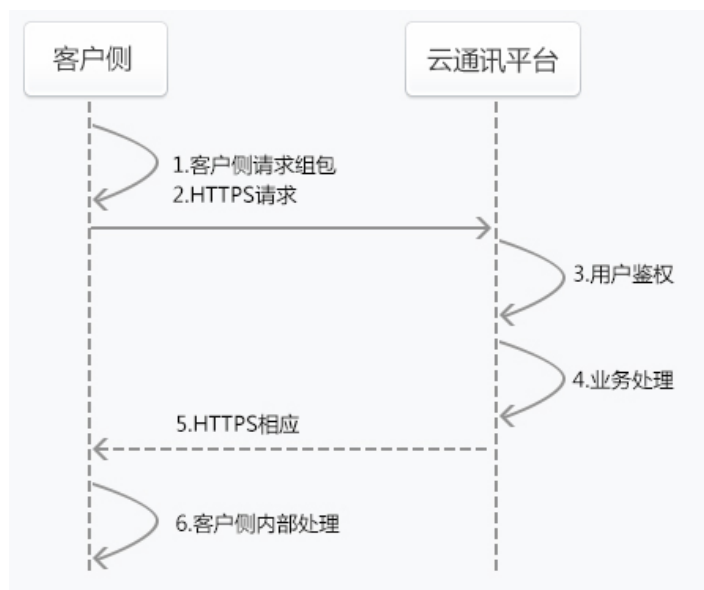
1 概述

云通讯平台作为通讯能力的云计算PAAS平台，将传统电信网络的通讯能力、基于IP的通讯能力，通过开放API及SDK的方式提供给开发者和商家，协助开发者快速、高效、低成本打造融合通讯能力的产品。本文档旨在为第三方应用开发者在PHP环境下实现云通讯平台REST API提供参考，文档预期的读者为第三方应用开发人员、平台开发人员、相关技术人员等。

1.1 介绍

云通讯平台REST API是为开发者提供的一套通讯能力API。通过它可以管理账户、电话、短信，也可以实现语音外呼、双向回呼、发送短信等功能。本文目的是通过实例介绍REST API基本功能的使用方法，尽量降低开发门槛，加快开发进度，缩短开发周期，将着重介绍REST API中创建子账户、双向回呼、发送短信等功能。

1.2 开发总体流程



业务流程详解：

1. 客户侧请求组包：根据具体业务需求构造请求包。
2. HTTPS请求：请求发给云通讯平台。
3. 用户鉴权：验证用户信息。
4. 业务处理：由云通讯平台实现请求内容。
5. HTTPS响应：返回执行结果。
6. 客户侧内部处理：客户得到执行结果后的业务处理。

1.3 安全机制

采用HTTP基本认证不是一种安全的认证方式，因为Base64编码仅仅是编码，而不是加密，以这种方式在互联网上传递用户名和密码，其危险性是显而易见的，所以建议与SSL一起使用以弥补HTTP基本认证在安全性方面的不足。应用服务器和平台之间采用的是双向认证，具体如下：



- SSL

SSL协议是与应用层协议独立无关的，建立在可靠的传输协议(例如：TCP)之上。高层的应用层协议(例如：HTTP)能透明的建立于SSL协议之上。SSL协议在应用层协议通信之前就已经完成加密算法、通信密钥的协商以及服务器认证工作。在此之后应用层协议所传送的数据都会被加密，从而保证通信的私密性。

- 应用服务器 → 云通讯平台

应用服务器发起REST API请求时，将账户Id (AccountSid)后追加一个英文冒号(:)然后串接上时间戳，得出的结果字符串再用Base64编码。例如，账户：rest，密码：123456，拼接后的结果是rest:123456，然后再用Base64编码字符串，编码后的字符串将附加于请求头Request Header中，同时在请求URL后附加参数sig，sig参数是账户Id (accountSid)、账户授权令牌(authToken)和时间戳的MD5加密大写串，平台在每次收到请求头后，根据协议取得应用服务器附加的认证信息，解开请求头，并对账户Id、账户授权令牌和时间戳进行验证，如果账户Id、账户授权令牌和时间戳都正确，则根据应用服务器的请求，平台返回所需要的数据；否则，返回错误代码或重新要求应用服务器提供账户Id、账户授权令牌和时间戳。

- 云通讯平台 → 应用服务器

平台使用HTTPS协议和应用服务器进行通信，每次访问应用服务器时，首先应用服务器要求平台提供MAC地址和AuthToken，然后平台在MAC后追加一个冒号(:)串接上AuthToken，得出的结果字符串再用Base64编码。例如：MAC:EC-55-F9-C7-62-CE，AuthToken:123456，拼接结果是EC-55-F9-C7-62-CE:123456，然后用BASE64编码字符串，将编码后的字符串附加于请求头Request Header中，应用服务器在每次收到请求头后，根据协议取得平台附加的认证信息，解开请求头，并对MAC、AuthToken进行验证，如果MAC、AuthToken都正确，则根据平台的请求，返回所需要的数据；否则，返回错误代码或重新要求提供MAC、AuthToken。

1.4 平台术语

- AS: Application Server，应用服务器，第三方开发者搭建的服务器，和云通讯平台交互，可以查询管理账户，也可以拨打电话、回拨电话、发送短信等。
- Base64: 网络上最常见的用于传输8Bit字节代码的编码方式之一，可用于在HTTP环境下传递较长的标识信息。
- CCP: Cloud Communication Platform，云通讯平台。
- CCP SDK: CCP Software Development Kit，云通讯平台软件开发包。
- MD5: Message Digest Algorithm MD5，消息摘要算法第五版，为计算机安全领域广泛使用的一种散列函数，用以提供消息的完整性保护。

- QML: Quick Markup Language, 快速标记语言, 一组当接收到来电或短信时告诉云通讯平台如何处理的指令。
- Rest: REpresentational State Transfer, 表征状态转移, 是一种针对网络应用的设计和开发方式, 可以降低开发的复杂性, 提高系统的可伸缩性。
- Rest服务器: 为应用服务器提供功能接口的服务器。
- VoIP: Voice over Internet Protocol, 基于网络协议的语音实时传输。
- VoIP帐号: 由VoIP服务器为子帐号分配的帐号。
- 开发者: 特指云通讯平台应用的第三方开发者。
- 主账号: 第三方开发者在云通讯平台开发者网站上注册后分配得到的账号。
- 子账号: 第三方开发者可使用主账号调用REST接口获取的账号。

1.5 参考文档

- 《云通讯平台REST 技术文档》

2 快速体验

- 通过下载CCP_REST_DEMO_PHP.rar, 解压后复制到开发者的PHP运行环境中, 通过浏览器访问即可方便的运行体验REST API。

2.1 申请测试账号

- 在云通讯平台进行注册, 注册之后创建Demo, 即可获得开发VoIP所需的测试帐号信息。
- 测试账号信息内容有: 主账号、主账号密码、子账号、子账号密码、VoIP账号、VoIP账号密码, 应用ID。

2.2 环境搭建

2.2.1 下载

- PHP5及以上版本, [下载](#)。
- APACHE2及以上版本, [下载](#)。

2.2.2 安装与设置

为了给开发者提供全面的技术支持, 在此以windows为例介绍PHP+Apache的环境搭建, 对于已有开发环境的开发者可以跳过此节。

1. 安装apache时默认安装, Network Domain、Server Name 填写开发者域名, Administrator's Email Address填写邮件地址。
2. 安装完后在安装目录下找到conf文件夹, 打开httpd.conf文件进行配置:
 - 找到 DocumentRoot, 将其设置为您所需存放php、htm等网页文件的web目录(如: "c:\Apache2.2\htdocs")。
 - 找到 DirectoryIndex, 在index.html后添加index.php、index.htm等, 以单个空格将其分开。
3. 将php的zip文件解压, 把解压的 php-X.X.X-Win32重命名为php并复制到C盘根目录下即安装路径为 C:\php。
4. 在php目录下找到php.ini-recommended文件重命名为php.ini并将其复制到系统所在目录(如: XP的Windows/system32目录)进行配置:
 - 将extension_dir 改为php/ext所在目录(如"C:\php\ext"), 表示指定PHP扩展包的具体目录, 以便调用相应的DLL文件。
 - 将doc_root 改为apache中同样的web目录(如"C:\Apache2.2\htdocs")。
 - 找到 ;session.save_path = "/tmp", 将';'去掉并设置开发者保存session的目录(如session.save_path = "C:/php/session")。
 - 如果需要支持Mysql和PHPMyAdmin需要将下面几句前面的分号去掉";"。

```
;extension=php_mbstring.dll
;extension=php_gd2.dll
;extension=php_mysql.dll
```

5. 最后增加apache的php支持。在http.conf文件中的#LoadModule vhost_alias_module modules/mod_vhost_alias.so下添加如下:

```
# 根据使用的apache版本选择DLL文件
LoadModule php5_module "c:/php/php5apacheX_X.dll"
PHPIniDir "C:/php"
AddType application/x-httpd-php .php .html .htm
```

6. 重启apache后在设置的DocumentRoot目录下创建php测试文件index.php, 填写内容<?php phpinfo();?>, 然后在浏览器中输入http://localhost, 就可以看到PHP的具体配置页面, 环境搭建完毕。

说明:

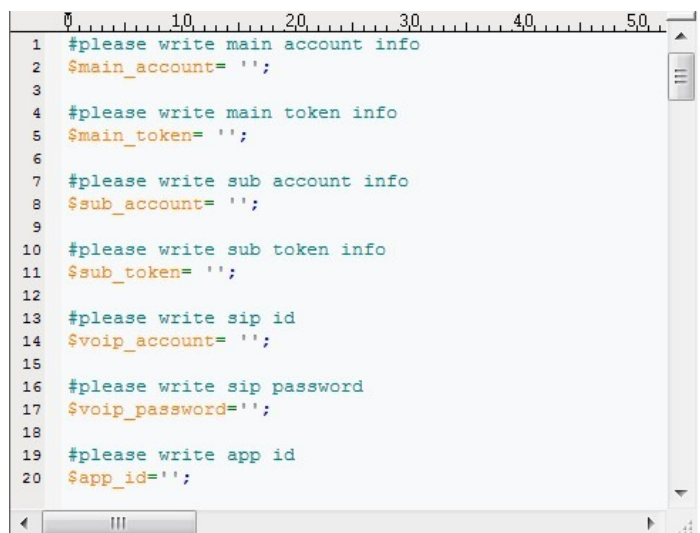
1. 如需快速搭建PHP环境, 可使用EasyPHP等集成了Apache+Mysql+Perl/PHP/Python的开发包, [下载](#)。
2. 本安装设置说明所涉及到的本地目录可由开发者自行调整。

2.3 Demo介绍

- 在Demo压缩包中包含REST_DEMO.php和REST_API.php两个文件。前者为REST API调用示例文件, 后者为REST API文件。
- REST API包含创建子账号, 双向回呼, 发送短信等功能接口。

2.4 配置帐号信息

- 打开REST_DEMO.php文件, 将申请测试账号时获取的Demo账号信息, 依次输入配置项中, 如图所示:



```
1 #please write main account info
2 $main_account= '';
3
4 #please write main token info
5 $main_token= '';
6
7 #please write sub account info
8 $sub_account= '';
9
10 #please write sub token info
11 $sub_token= '';
12
13 #please write sip id
14 $voip_account= '';
15
16 #please write sip password
17 $voip_password='';
18
19 #please write app id
20 $app_id='';
```

2.5 运行体验

将REST_DEMO.php和REST_API.php复制到web目录，需要通过浏览器运行。下面对API的使用做如下说明：

API基本函数：

- HTTPS请求函数

```
function curl_post($url,$data,$header,$post=1){  
    // 初始化curl  
    $ch = curl_init();  
    // 参数设置  
    $res=curl_setopt($ch, CURLOPT_URL,$url);  
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE);  
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);  
    curl_setopt($ch, CURLOPT_HEADER, 0);  
    curl_setopt($ch, CURLOPT_POST, $post);  
    if( $post)  
        curl_setopt($ch, CURLOPT_POSTFIELDS, $data);  
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);  
    curl_setopt($ch,CURLOPT_HTTPHEADER,$header);  
    $result = curl_exec($ch);  
    // 出错则显示错误信息  
    if($result == FALSE){  
        print curl_error($ch);  
    }  
    curl_close($ch);  
    return $result;  
}
```

2.5.1 创建子账户

功能：云通讯平台收到请求后，返回子账户信息，同时返回Voip账号信息。

用法：调用创建子账户时先初始化REST

API，然后调用CreateSubAccount()函数，开发者需要填入待创建的子账户名称、子账户类型、子账户状态。

API函数详解

1. 函数组包:本函数需要使用主账号鉴权，sig参数和包头中的时间戳必须一致。

- 构造请求URL内容：首先获得系统时间戳；再拼接sig字符串，格式为“主账户id + 主账户授权令牌 + 时间戳”；然后使用MD5加密sig字符串；最后拼接URL内容
- 构造请求头信息：设置Accept项为“application/xml”，如果是json交互方式，请设置为“application/json”；Content-Type项为“application/xml;charset=utf-8”，如果是json交互方式，请设置为“application/json;charset=utf-8”；Authorization项设置时，先拼接字符串“主账户id + 冒号 + 时间戳”，冒号为英文格式，再将字符串进行Base64编码
- 构造Body：拼接XML或者Json格式字符串，字段包括应用id、子账户名称

```
/* 全局变量
 * this->app_id 应用id
 * this->main_account 主账户id
 * this->main_token 主账户授权令牌
 * this->batch 时间戳
 * this->address 服务器地址
 * this->soft_version REST API版本
 */
{ // 拼接请求包体
    $body = "";
    if ($this->sendType != 0) // json方式
        $body="{ 'appld': '$this->app_id', 'friendlyName': '$friendlyName' }";
    else // xml方式
        $body="<SubAccount>
            <appld>$this->app_id</appld>
            <friendlyName>$friendlyName</friendlyName>
        </SubAccount>";
    // 大写的sig参数
    $sig=strtoupper(md5($this->main_account . $this->main_token . $this->batch));
    // 生成请求URL
    $url=https://$this->address/$this->soft_version/Accounts/$this->main_account/SubAccounts?sig=$sig;
    // 生成授权：主账户id + 英文冒号 + 时间戳。
    $authen=base64_encode($this->main_account . ":" . $this->batch);
    // 生成包头 $this->dataType是设置好的字符串变量，xml方式交互时是"xml"，json交互时是"json"
    $header =
    array("Accept:application/$this->dataType","Content-Type:application/$this->dataType;charset=utf-8","Authorization:$authen");
}
```

2. 请求与解析：调用curl封装函数发送HTTPS请求，得到结果为XML或者Json格式数据。

```
{  
    // 发送HTTPS请求  
    $result=curl_post($url,$body,$header);  
}
```

实例调用

返回时可以通过\$data->statusCode获取状态码，成功返回值为0。返回正确时可以通过\$data->SubAccount获取返回信息

```
{  
    // 创建REST对象实例  
    $rest=new REST($main_account,$main_token,$app_id);  
    // 调用云通讯平台的创建子账号,填入需要绑定的子账户名称  
    $result=$rest->CreateSubAccount($friendlyName,$status,$type);  
    if($result == NULL){  
        echo "result error!";  
        break;  
    }  
    $data = "";  
    if ($sendType == 0){  
        // 解析xml  
        $data = simplexml_load_string(trim($result," \t\n\r"));  
    }  
    else{  
        // 解析json  
        $data = json_decode(trim($result," \t\n\r"));  
    }  
    if($data->statusCode!=0){  
        echo "error code :". $data->statusCode . "<br/>";  
        //TODO 添加错误处理逻辑  
    }  
    else{  
        echo "create SubbAccount success<br/>";  
        // 获取返回信息  
        $subaccount = $data->SubAccount;  
        echo "subAccountid:". $subaccount->accountSid . "<br/>";  
        ...  
        echo "type:". $subaccount->type . "<br/>";  
        //TODO 把云平台创建账号信息存储在您的服务器上.  
        //TODO 添加成功处理逻辑  
    }  
}
```


2.5.2 发送短信

功能：云通讯平台收到请求后，向一个有短信能力的终端发送短信。

用法：调用发送短信时先初始化REST API，然后调用SendSMS()函数，开发者需要填入接收端手机号码集合、短信内容、消息类型、子账户id。

API函数详解

1. 函数组包:本函数需要使用主账号鉴权，sig参数和包头中的时间戳必须一致。

- 构造请求URL内容：首先获得系统时间戳；再拼接sig字符串，格式为“主账户id + 主账户授权令牌 + 时间戳”，然后使用MD5加密sig字符串，最后拼接URL内容
- 构造请求头信息：设置Accept项为“application/xml”，如果是json交互方式，请设置为“application/json”；Content-Type项为“application/xml;charset=utf-8”，如果是json交互方式，请设置为“application/json;charset=utf-8”；Authorization项设置时，先拼接字符串“主账户id + 冒号 + 时间戳”，冒号为英文格式，再将字符串进行Base64编码
- 构造Body：拼接XML或者Json格式字符串，字段包括接收端手机号码集合、短信正文、消息类型、应用id、子账户id

```
/* 全局变量
 * this->app_id 应用id
 * this->main_account 主账户id
 * this->main_token 主账户授权令牌
 * this->batch 时间戳
 * this->address 服务器地址
 * this->soft_version REST API版本
 */
{
    // 拼接请求包体
    if ($this->sendType != 0)
        $body=
        '{"to':$to,'body': '$body','msgType': '$msgType','appId': '$this->app_id','subAccountSid': '$sub_account'}';
    else
        $body="<SMSMessage>
        <to>$to</to>
        <body>$body</body>
        <msgType>$msgType</msgType>
        <appId>$this->app_id</appId>
        <subAccountSid>$sub_account</subAccountSid>
        </SMSMessage>";
    // 大写的sig参数
    $sig=strtoupper(md5($this->main_account . $this->main_token . $this->batch));
    // 生成请求URL
    $url=https://$this->address/$this->soft_version/Accounts/$this->main_account/SMS/Messages?sig=$sig;

    // 生成授权: 主账户id + 英文冒号 + 时间戳。
    $authen=base64_encode($this->main_account . ":" . $this->batch);
    // 生成包头$this->dataType是设置好的字符串变量, xml方式交互时是"xml", json交互时是"json"
    $header =
    array("Accept:application/$this->dataType","Content-Type:application/$this->dataType;charset=utf-8","Authoriz
    ation:$authen");
}
```

2. 请求与解析: 调用curl封装函数发送HTTPS请求, 得到结果为XML或者Json格式数据。

```
{
    // 发送HTTPS请求
    $result=curl_post($url,$body,$header);
}
```

实例调用

返回时可以通过\$data->statusCode获取状态码，成功返回值为0。返回正确时可以通过\$data->SMSMessage获取返回信息

```
{
    // 创建REST对象实例
    global $main_account,$main_token,$app_id;
    $rest=new REST($main_account,$main_token,$app_id);
    // 发送短信
    $rest->SendSMS($to,$body,$msgType,$sub_account);
    if($result == NULL){
        echo "result error!";
        break;
    }
    $data = "";
    if($sendType == 0){
        // 解析xml
        $data = simplexml_load_string(trim($result," \t\n\r"));
    }
    else{
        // 解析json
        $data = json_decode(trim($result," \t\n\r"));
    }
    if($data->statusCode!=0){
        echo "error code:". $data->statusCode . "<br>";
        //TODO 添加错误处理逻辑
    }
    else{
        echo "Sendind message success!<br/>";
        // 获取返回信息
        $smsmessage = $data->SMSMessage;
        echo "dateCreated:".$$smsmessage->dateCreated."<br/>";
        echo "smsMessageSid:".$$smsmessage->smsMessageSid."<br/>";
        //TODO 添加成功处理逻辑
    }
}
```

说明:

发送短信后，根据应用的状态决定用户能否发送短信：

- 如果应用状态为禁用(0)或者删除(-1)，则不能发送短信，
- 如果应用状态为上线(2)，则能发信息给任意手机号，
- 如果应用状态为未发布/下线(1)、审核中(3)、审核未通过(4)，则只能发信息给注册手机号，短信发送成功则进行短信计费。

2.5.3 双向回呼

功能：云通讯平台收到请求后，向两个落地电话终端发起呼叫请求，两终端接通电话后进行通话。

用法：调用双向回呼时先初始化REST API，然后调用CallBack()函数，开发者需要填入主叫号码、被叫号码、VoIP账号、子账户id、子账号密码。

API函数详解

1.函数组包:本函数需要使用子账号鉴权，sig参数和包头中的时间戳必须一致。

- 构造请求URL内容：首先获得系统时间戳；再拼接sig字符串，格式为“子账户id + 子账户授权令牌 + 时间戳”，然后使用MD5加密sig字符串，最后拼接URL内容
- 构造请求头信息：设置Accept项为“application/xml”，如果是json交互方式，请设置为“application/json”；Content-Type项为“application/xml;charset=utf-8”，如果是json交互方式，请设置为“application/json;charset=utf-8”；Authorization项设置时，先拼接字符串“子账户id + 冒号 + 时间戳”，冒号为英文格式，再将字符串进行Base64编码
- 构造Body：拼接XML或者Json格式字符串，字段包括子账户id、主叫号码、被叫号码

```
/* 全局变量
* this->app_id 应用id
* this->main_account 主账户id
* this->main_token 主账号授权令牌
* this->batch 时间戳
* this->address 服务器地址
* this->soft_version REST API版本
*/
{
    // 拼接请求包体
    if ($this->sendType != 0)
        $body= "{from:'$from','to':'$to'}";
    else
        $body= "<CallBack>
<subAccountSid>$sub_account</subAccountSid>
<from>$from</from>
<to>$to</to>
</CallBack>";
    // 大写的sig参数
    $sig=strtoupper(md5($sub_account . $sub_token . $this->batch));
    // 生成请求URL
    $url=https://$this->address/$this->soft_version/SubAccounts/$sub_account/Calls/Callback?sig=$sig;
    // 生成授权：子账户id + 英文冒号 + 时间戳。
    $authen=base64_encode($sub_account . ":" . $this->batch);
    // 生成包头$this->dataType是设置好的字符串变量，xml方式交互时是"xml"，json交互时是"json"
    $header =
    array("Accept:application/$this->dataType","Content-Type:application/$this->dataType;charset=utf-8","Authorization:$authen");
}
```

2. 请求与解析: 调用curl封装函数发送HTTPS请求, 得到结果为XML或者Json格式数据。

```
{  
    // 发送HTTPS请求  
    $result=curl_post($url,$body,$header);  
}
```

实例调用

返回时可以通过\$data->statusCode获取状态码, 成功返回值为0。返回正确时可以通过\$data->CallBack获取返回信息

```
{  
    // 创建REST对象实例  
    global $main_account,$main_token,$app_id,$sub_account,$sub_token,$voip_account;  
    $rest = new REST($main_account,$main_token,$app_id,$sendType);  
    // 调用回拨接口  
    echo "Try to make a callback,called is $to <br/>";  
    $result = $rest->CallBack($from,$to,$sub_account,$sub_token);  
    if($result == NULL){  
        echo "result error!";  
        break;  
    }  
    $data = "";  
    if($sendType == 0){  
        // 解析xml  
        $data = simplexml_load_string(trim($result," \t\n\r"));  
    }  
    else{  
        // 解析json  
        $data = json_decode(trim($result," \t\n\r"));  
    }  
    if($data->statusCode!=0){  
        echo "error code :". $data->statusCode . "<br>";  
        //TODO data  
    }  
    else{  
        echo "callback success!<br>";  
        // 获取返回信息  
        $callback = $data->CallBack;  
        echo "callSid:".$callback->callSid."<br/>";  
        echo "dateCreated:".$callback->dateCreated."<br/>";  
        //TODO 添加成功处理逻辑  
    }  
}
```

3 错误码

3.1 云通讯平台错误码

请参考《平台错误码》

3.2 HTTPS标准错误码

100:继续。	100 Continue
101:切换协议。	101 Switching Protocols
200:ok	200 OK
201:已创建。	201 Created
202:已接受。	202 Accepted
203:非权威性信息。	203 Non-Authoritative Information
204:无内容。	204 No Content
205:重置内容。	205 Reset Content
206:部分内容。	206 Partial Content
300:多重选择。	300 Multiple Choices
301:永久删除。	301 Moved Permanently
302:临时删除。	302 Moved Temporarily
303:参照其他。	303 See Other
304:未修改。	304 Not Modified
305:使用代理。	305 Use Proxy
307:临时重定向。	307 Temporary Redirect

400:请求无效。	400 Bad Request
401:访问被拒绝。	401 Unauthorized
402:需要付费。	402 Payment Required
403:禁止访问。	403 Forbidden
404:无法找到文件。	404 Not Found
405:资源被禁止。	405 Method Not Allowed
406:无法接受。	406 Not Acceptable
407:要求代理身份验证。	407 Proxy Authentication Required
408:请求超时。	408 Request Timeout
409:冲突。	409 Conflict
410:永远不可用。	410 Gone
411:要求长度。	411 Length Required
412:先决条件失败。	412 Precondition Failed
413:请求实体太大。	413 Request Entity Too Large
414:请求:URI:太长。	414 Request-URI Too Long
415:不支持的媒体类型。	415 Unsupported Media Type
416:所请求的范围无法满足。	416 Requested Range Not Satisfiable
417:执行失败。	417 Expectation Failed
500:内部服务器错误。	500 Internal Server Error
501:未实现。	501 Not Implemented
502:网关错误。	502 Bad Gateway
503:服务不可用。	503 Service Unavailable

504:网关超时。	504 Gateway Timeout
505:HTTP:版本不受支持。	505 HTTP Version Not Supported