# 移动解析 HttpDNS

# API 文档

## 产品文档

腾讯云

# 文档目录

API 文档
- 免费版 API 接入
- 企业版API接入（支持加密）
- 企业版API Https接入
- 企业版/免费版API接入最佳实践
- 企业版SDK接入

# API 文档
# 免费版 API 接入

最近更新时间：2018-06-25 10:52:01

## 根据域名和用户 IP 查询

数据请求和应答均使用 HTTP 协议。
请求格式为：

```
http://119.29.29.29/d?dn=www.dnspod.cn.&ip=1.1.1.1
```

其中， `dn` 表示要查询的域名；
`ip` 表示用户 IP，当 `ip` 为内网 IP或非法 IP，默认取 HTTP 报文的源 IP 为用户 IP，此时 HttpDNS 会按照用户设置的默认线路进行解析。

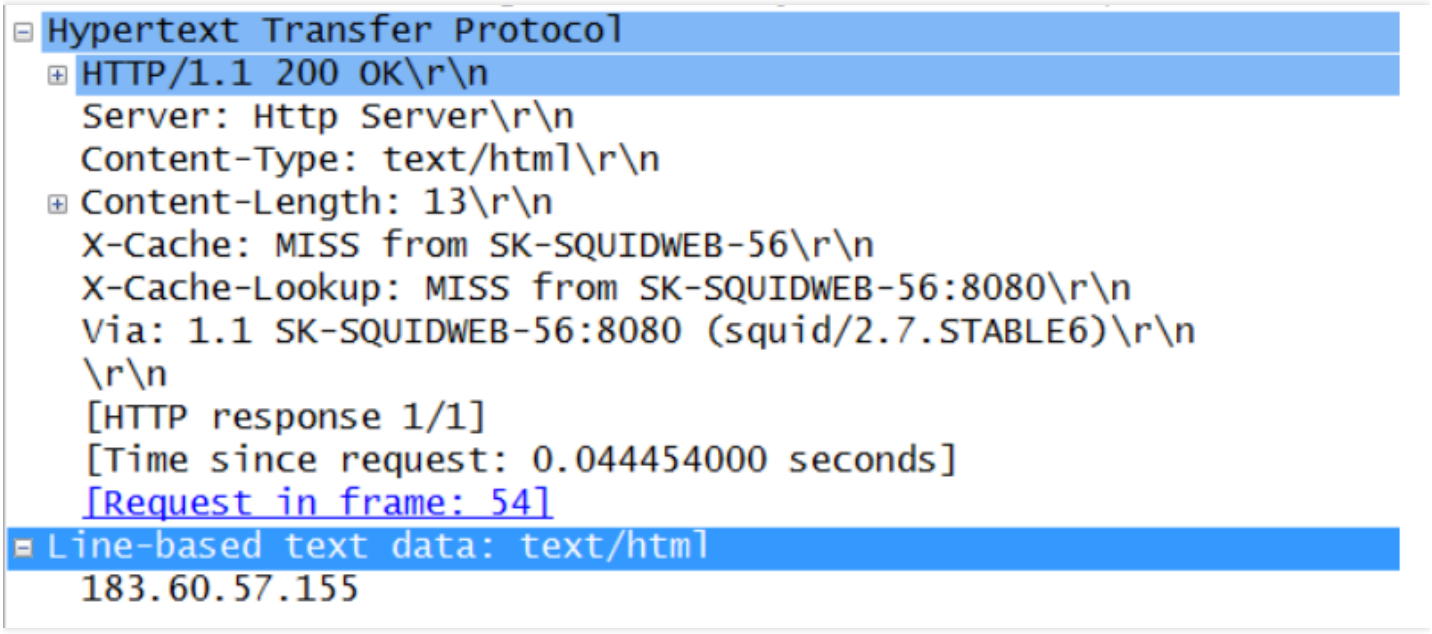
### 域名存在

如果 HttpDNS 能查询到最终的 IP 指向，则直接返回 IP。
以下列请求为例

```
http://119.29.29.29/d?dn=www.dnspod.cn.&ip=1.1.1.1
```

返回

```
183.60.57.155
```

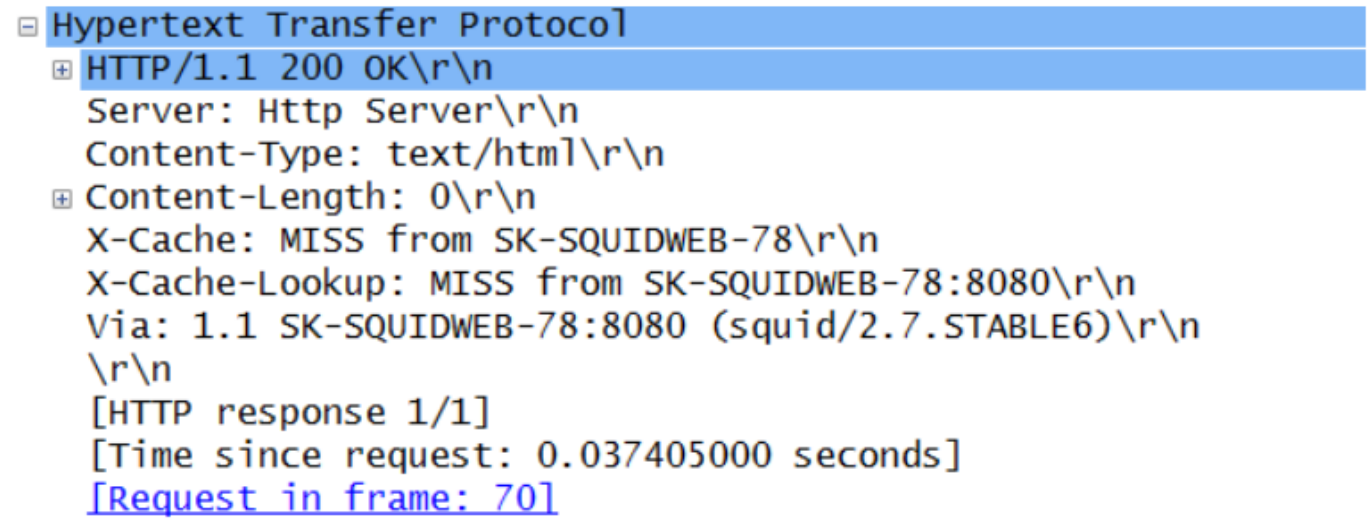



### 域名不存在

如果域名不存在，HttpDNS 无法查询到最终的IP指向，则返回空。
以下列请求为例：

```
http://119.29.29.29/d?dn=www.dnspod2.cn.&ip=1.1.1.1
```

返回空字符串，如下图所示：




## 根据域名和用户 IP 查询带 TTL 的结果

数据请求和应答均使用 HTTP 协议。
请求格式为:

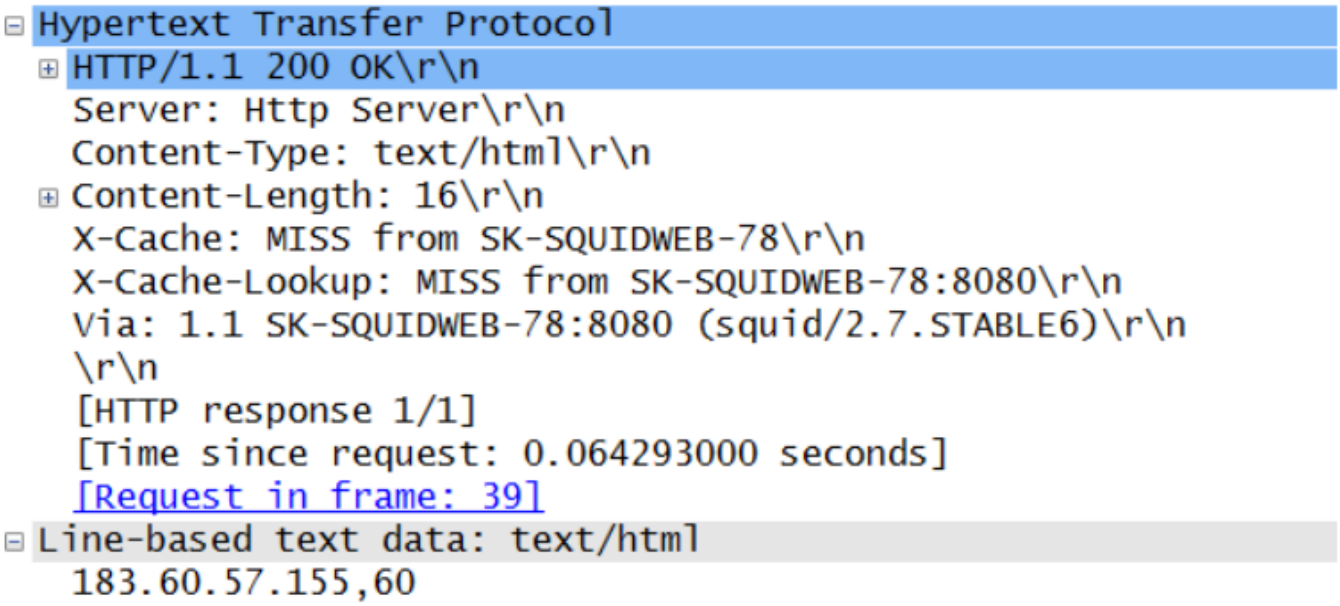
```
http://119.29.29.29/d?dn=www.dnspod.cn.&ip=1.1.1.1&ttl=1
```

`ttl=1` 表示要求返回结果携带解析结果的 TTL 值。
返回的 TTL 和域名解析结果用英文逗号分隔。

以下列请求为例，表示要求返回结果带上 TTL：

```
http://119.29.29.29/d?dn=www.dnspod.cn.&ip=1.1.1.1&ttl=1
```

返回值如下，表示递归服务器缓存的 TTL 是 60 秒：
`183.60.57.155,60` ：



# 企业版API接入（支持加密）

最近更新时间：2018-06-04 17:51:58

Des 加密功能可以防止明文 HTTP 请求在传输过程中被恶意篡改，使用加密功能需申请企业版本。具体使用方式如下：

## 1. 开通企业版本

仅 HttpDNS 企业版本可以使用加密功能，将收到授权 ID 和密钥，请妥善保管、切勿泄漏。

## 2. 基本步骤

第一步：查收您邮件中的授权 ID 和对应的密钥；
第二步：将查询的域名（如客户指定用 IP，则 IP 需要加密）用授权 ID 和密钥，以 Des 的 ECB 方式进行加密，填充方式为 PKCS5Padding。
第三步：发送加密的请求；
第四步：接受加密的应答；
第五步：将结果解密，最终获得所查询的域名对应的解析结果。

下面以 Android 为例说明：

## 3. Android 示例

### 3.1 加密发送请求

先将要查询的域名用您收到的邮件中的ID对应的密钥，以 Des 的 ECB 方式进行加密，填充方式为 PKCS5Padding。如果需要指定 IP 参数，IP 参数也用同样的方法加密。

```
try {
//初始化密钥
SecretKeySpec keySpec = new SecretKeySpec(encKey.getBytes("utf-8"), "DES");
//选择使用DES算法，ECB方式，填充方式为PKCS5Padding
Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
//初始化
cipher.init(Cipher.ENCRYPT_MODE, keySpec);
//获取加密后的字符串
encryptedString = bytesToHex(cipher.doFinal(hostName.getBytes("utf-8")));
} catch (Exception e) {
```

```
e.printStackTrace();
}
```

### 3.2 发送请求

域名加密后，向 HttpDNS 服务器发起请求：

```
//dn参数对应为加密后的字符串，id 对应为您的密钥 ID
dn=ac7875d400dacdf09954edd788887719&ip=30958d601665478905668b8556976250&id=1&ttl=1
```

### 3.3 接受加密应答

当您将加密后的请求发送给 HttpDNS 后，客户端将会收到一串加密后的结果：

```
`60a111ecb44008ac1b32d1fdfb42aa8a96bade20444421dcf83362072c84cf2ad8f870dfb0a1e448`
```

### 3.4 结果解密

```
try {
//初始化密钥
SecretKeySpec keySpec = new SecretKeySpec(encKey.getBytes("utf-8"), "DES");
//选择使用 DES 算法，ECB 方式，填充方式为 PKCS5Padding
Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
//初始化
cipher.init(Cipher.DECRYPT_MODE, keySpec);
//获取解密后的字符串
decryptedString = cipher.doFinal(hexToBytes(s));
} catch (Exception e) {
e.printStackTrace();
}
```

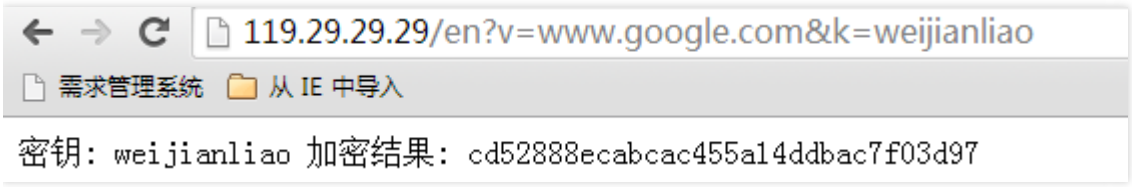
至此，您已获得解密后的域名解析结果。

## 4. 加解密测试

如果需要进行测试，可以使用以下加密及解密测试功能（此功能仅针对已申请企业版的用户开放）：

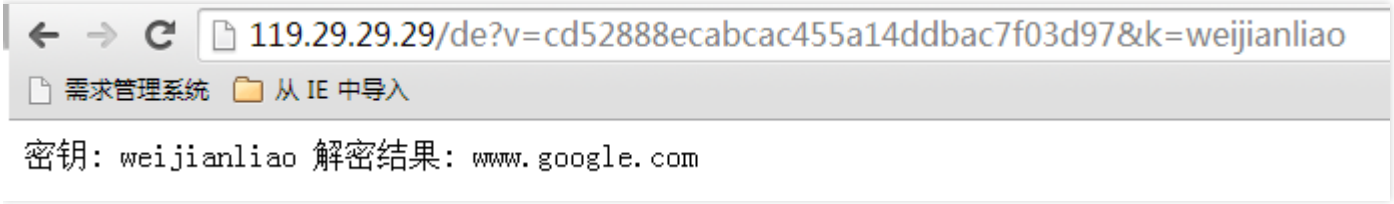
加密：

```
http://119.29.29.29/en?v=www.google.com&k=weijianliao
```




解密：

```
http://119.29.29.29/de?v=cd52888ecabcac455a14ddbac7f03d97&k=weijianliao
```




### 附：iOS示例

**MSDKDnsInfoTool.h**

```
//
// MSDKDnsInfoTool.h
// MSDKDns
//
// Created by Mike on 3/25/16.
// Copyright © 2016 Tencent. All rights reserved.
//

#import <Foundation/Foundation.h>

@interface MSDKDnsInfoTool : NSObject

+ (NSString *) encryptUseDES:(NSString *)plainText key:(NSString *)key;
+ (NSString *) decryptUseDES:(NSString *)cipherString key:(NSString*)key;

@end
```

**MSDKDnsInfoTool.m**

```
//
// MSDKDnsInfoTool.m
// MSDKDns
```

```
//
// Created by Mike on 3/25/16.
// Copyright © 2016 Tencent. All rights reserved.
//

#import "MSDKDnsInfoTool.h"
#import <CommonCrypto/CommonDigest.h>
#import <CommonCrypto/CommonCrypto.h>

@implementation MSDKDnsInfoTool

char MSDKDnsByteToHexByte(char byte) {
if (byte < 10) {
return byte + '0';
}
return byte - 10 + 'a';
}

void MSDKDnsByteToHexChar(char byte, char *hex) {
hex[0] = MSDKDnsByteToHexByte((byte >> 4) & 0x0F);
hex[1] = MSDKDnsByteToHexByte(byte & 0x0F);
}

NSString * MSDKDnsDataToHexString(NSData *data) {
char hex[data.length * 2 + 1];
const char *bytes = (const char *)data.bytes;
for (NSUInteger i = 0; i < data.length; ++i) {
MSDKDnsByteToHexChar(bytes[i], &hex[i * 2]);
}
hex[data.length * 2] = 0;
return [NSString stringWithUTF8String:hex];
}

char MSDKDnsHexByteToChar(char hex) {
if (hex >= '0' && hex <= '9') {
return hex - '0';
}
if (hex >= 'a' && hex <= 'f') {
return hex - 'a' + 10;
}
if (hex >= 'A' && hex <= 'F') {
return hex - 'A' + 10;
}
return 0;
}
```

```
char MSDKDnsHexCharToChar(char high, char low) {
high = MSDKDnsHexByteToChar(high);
low = MSDKDnsHexByteToChar(low);
return (high << 4) | low;
}

+ (NSString *) encryptUseDES:(NSString *)plainText key:(NSString *)key {
NSData *srcData = [plainText dataUsingEncoding:NSUTF8StringEncoding];
size_t dataOutAvilable = ([srcData length] + kCCBlockSizeDES) & ~(kCCBlockSizeDES - 1);
unsigned char dataOut[dataOutAvilable];
memset(dataOut, 0x0, dataOutAvilable);
size_t dataOutMoved = 0;

char encryptKey[kCCKeySizeDES] = {0};
strncpy(encryptKey, [key UTF8String], kCCKeySizeDES);

CCCryptorStatus ccStatus = CCCrypt(kCCEncrypt,
kCCAlgorithmDES,
kCCOptionPKCS7Padding | kCCOptionECBMode,
encryptKey,
kCCKeySizeDES,
NULL,
srcData.bytes,
srcData.length,
dataOut,
dataOutAvilable,
&dataOutMoved);
if (ccStatus == kCCSuccess) {
NSData * resultData = [NSData dataWithBytes:dataOut length:(NSUInteger)dataOutMoved];
return MSDKDnsDataToHexString(resultData);
}
return nil;
}

+ (NSString *) decryptUseDES:(NSString *)cipherString key:(NSString *)key {
if (cipherString && key) {
const char *tempBytes = [cipherString UTF8String];
NSUInteger tempLength = [cipherString length];
if (tempLength > 0) {
NSUInteger dataLength = tempLength / 2;
char textBytes[dataLength];
for (int i = 0; i < tempLength - 1; i = i + 2)
{
char high = tempBytes[i];
```

```
char low = tempBytes[i + 1];
char hex = MSDKDnsHexCharToChar(high, low);
textBytes[i / 2] = hex;
}

size_t dataOutAvilable = (dataLength + kCCBlockSizeDES) & ~(kCCBlockSizeDES - 1);
unsigned char dataOut[dataOutAvilable];
memset(dataOut, 0x0, dataOutAvilable);
size_t dataOutMoved = 0;

char decryptKey[kCCKeySizeDES] = {0};
strncpy(decryptKey, [key UTF8String], kCCKeySizeDES);
CCCryptorStatus ccStatus = CCCrypt(kCCDecrypt,
kCCAlgorithmDES,
kCCOptionPKCS7Padding | kCCOptionECBMode,
decryptKey,
kCCKeySizeDES,
NULL,
textBytes,
dataLength,
dataOut,
dataOutAvilable,
&dataOutMoved);

NSString *plainText = nil;
if (ccStatus == kCCSuccess) {
NSData *data = [NSData dataWithBytes:dataOut length:(NSUInteger)dataOutMoved];
if (data) {
plainText = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
}
}
return plainText;
}
}
return nil;
}

@end
```

# 企业版API Https接入

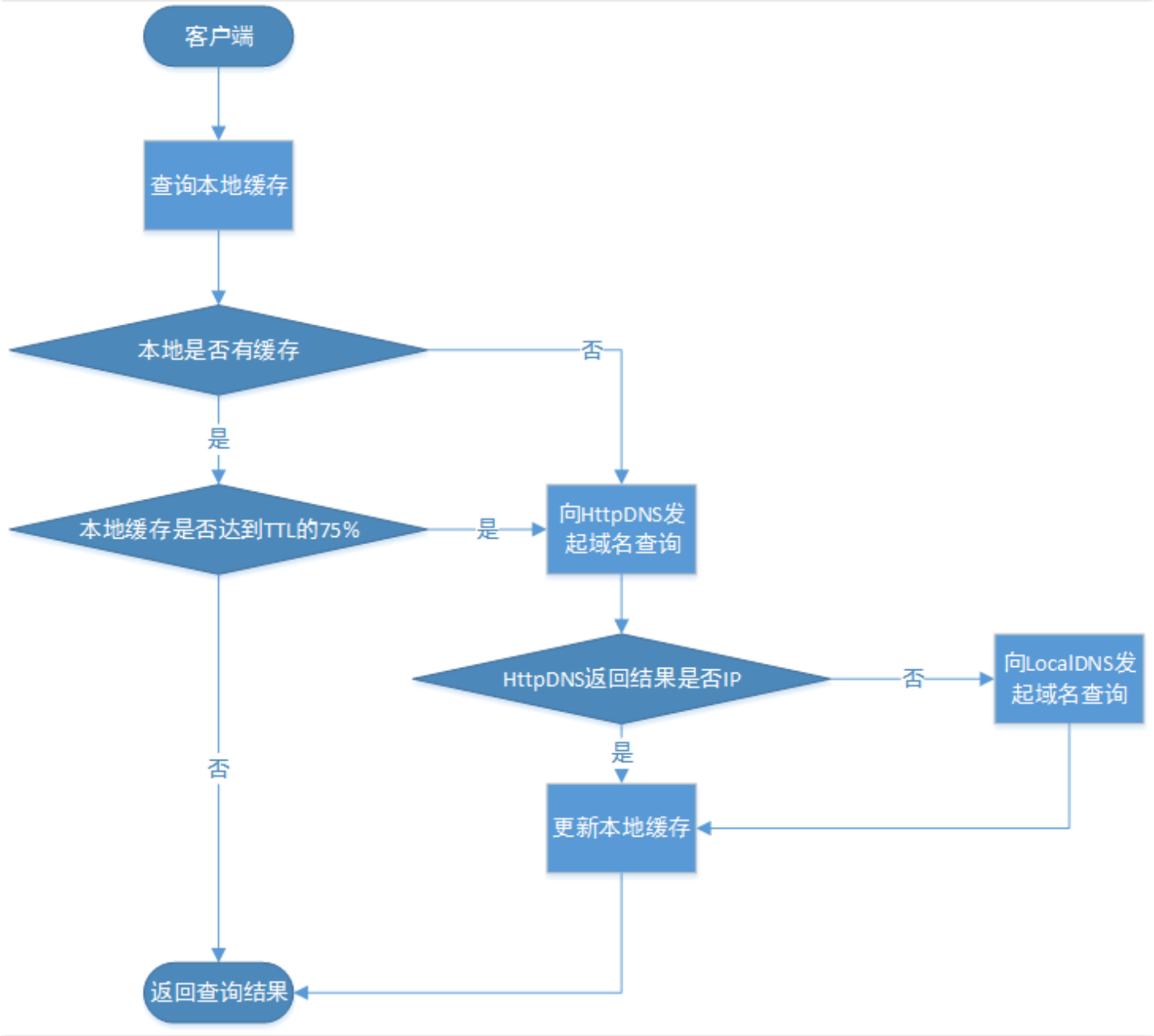
最近更新时间：2018-08-31 11:51:19

目前该服务仅对企业版提供 Https 接入申请，如需申请，请注明具体使用场景及预计调用频率 提交工单 联系我们。

# 企业版/免费版API接入最佳实践

最近更新时间：2018-09-27 11:43:54

接入HttpDNS过程中，需要改造移动APP的域名解析机制，新的流程参考如下：




改造过程中需要遵循以下两个设计策略：

## 1. Failed over策略

虽然HttpDNS已经接入BGP Anycast，并实现了多地跨机房容灾。但为了保证在最坏的情况下客户端域名解析依然不受影响。建议采用以下的fail over策略：

- 第一步先向HttpDNS发起域名查询请求
- 如果HttpDNS查询返回的结果不是一个IP地址（结果为空、结果非IP、连接超时等），则通过本地LocalDNS进行域名解析。超时时间建议为5s。

### 2. 缓存策略

移动互联网用户的网络环境比较复杂，为了尽可能地减少由于域名解析导致的延迟，建议在本地进行缓存。缓存规则如下：

- 缓存时间
  缓存时间建议设置为120s至600s，不可低于60s。

- 缓存更新
  缓存更新应在以下两种情形下进行：
  **用户网络状态发生变化时：**
  移动互联网的用户的网络状态由3G切Wi-Fi，Wi-Fi切3G的情况下，其接入点的网络归属可能发生变化。所以用户的网络状态发生变化时，需要重新向HttpDNS发起域名解析请求，以获得用户当前网络归属下的最优指向。
  **缓存过期时：**
  当域名解析的结果缓存时间到期时，客户端应该向HttpDNS重新发起域名解析请求以获取最新的域名对应的IP。为了减少用户在缓存过期后重新进行域名解析时的等待时间，建议在75%TTL时就开始进行域名解析。如本地缓存的TTL为600s，那么在第600*0.75=450s时刻,客户端就应该进行域名解析。

除了以上几点建议外，减少域名解析的次数也能有效的减少网络交互，提升用户访问体验。建议在业务允许的情况下，尽量减少域名的数量。如需区分不同的资源，建议通过url来进行区分。

### 3. 其他注意事项

改造APP中的需要关注的tips：

1. 请尽量将不同功能用同样域名，资源区分通过url来实现，减少域名解析次数（用户体验好，容灾切换方便。多一个域名，即使域名已命中缓存，至少多100ms的访问延迟），新版本将很快支持批量域名解析
2. 设置的缓存TTL值不可太低（不可低于60s），防止频繁进行HtppDNS请求。
3. 接入HttpDNS的业务需要保留用户本地LocalDNS作为容灾通道，当HttpDNS无法正常服务时（移动网络不稳定或HttpDNS服务出现问题），可以使用LocalDNS进行解析，。
4. 安卓程序中可能出现404错误，但浏览器中正常。可能为权限问题，或其他问题，参考 http://stackoverflow.com/questions/10835845/android-http-request-wierd-404-not-found-issue
5. bytetohex&hextobyte，需自己实现接口，进行16进制字符串与字节的转换
6. https问题，需在客户端hook客户端检查证书的domain域和扩展域看是否包含本次请求的host的过程，将IP直接替换成原来的域名，再执行证书验证。或者忽略证书认证，类似于curl -k参数。
7. HttpDNS请求建议超时时间2-5s左右。
8. 在网络类型变化时，如4G切换到wifi，不同wifi间切换等，需要重新执行HttpDNS请求刷新本地缓存。

# 企业版SDK接入

最近更新时间：2017-09-26 11:53:52

企业版本用户，官方提供腾讯自研 **智营SDK** ，定制化、可直接嵌入APP内调用，已经广泛应用于腾讯各类游戏客户端，功能成熟稳定。

具体可参考以下文档：

iOS版本 SDK >>

Android版本 SDK >>