ALIBABA CLOUD

阿里云

API文档

文档版本：20220705

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.概述 05
2.DNS over HTTPs(DoH) 06
3.DoH JSON API 10
4.DNS over TLS （DoT） 15
5.Android SDK开发指南 17
6.iOS SDK开发指南 31

1.概述

阿里公共DNS致力于为广大的互联网用户提供快速、稳定和安全的DNS解析。传统的DNS查询和应答采用UDP和TCP明文传输，存在网络监听、DNS劫持、中间设备干扰的风险：

- 网络监听风险：即便用户采用HTTPS加密的方式访问站点，DNS查询应答并没有采用加密传输
- DNS劫持：传统DNS应答数据会被篡改，用户的访问会被路由到钓鱼网站和恶意站点
- 中间设备干扰：主要包括防火墙的拦截或篡改，针对域名的过滤，以及大包MTU分片等

为了应对以上挑战，阿里公共DNS遵守DoH（RFC8484）和DoT(RFC7858) 标准对外提供DNS的安全传输服务，支持DNS over HTTPs，DNS over TLS两种安全传输模式，以及基于 HTTP 和 HTTPS 的 DoH JSON API 接口。DNS的安全传输服务可以适用于移动应用程序、浏览器、操作系统、物联网设备，网关和路由器等多个场景。通过传输加密的方式发送DNS查询，加强了用户访问互联网的安全性、解析稳定和隐私保护。

除了隐私加密以外，用户端和DNS服务器间可以使用TCP或HTTP连接来提供安全传输服务，一方面可以服务精准的基于位置的DNS解析和流量调度，另一方面基于DNS端到端的连接特性，DNS的动态变更可以实现秒级端到端生效。

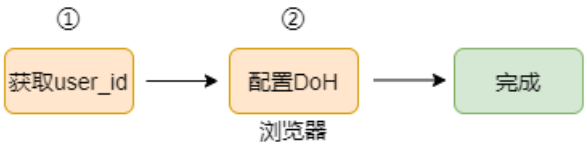
注意：在DoH和DoT传输服务中，阿里公共DNS支持 TLS 1.2 和TLS 1.3。

2.DNS over HTTPs(DoH)

本章节介绍DoH的接入过程。

阿里公共DNS通过RFC 8484指定的经过TLS加密的HTTP连接提供DNS解析

接入流程如下：



https://user_id.alidns.com/dns-query

获取user_id

请在公共DNS控制台查看account_id。如下：

普通用户配置DoH

假设account_id是9999，用户的DoH接口配置为：

https://9999.alidns.com/dns-query

接下来，将以**Firefox（火狐浏览器）**为例，介绍如何配置DoH。

第一步：从 Firefox 菜单栏中点击 【选项】设置。

第二步：搜索框中输入“DNS”，点击【网络设置】中的【设置】按钮。

第三步：在设置页面的底部，找到并勾选“启用基于HTTPS的DNS”，选择“自定义”的提供商，并输入接口地址：https://9999.alidns.com/dns-query，点击【确定】。

开发者使用DoH服务

对开发者，需要按照RFC8484中的标准要求生成DoH的 HTTP request请求发送给DoH服务地址。

DoH可以同时用GET和POST两种方法。当使用POST方法，DNS查询消息被包含在HTTP 请求的message body中。当使用GET方法，唯一的变量"dns"被赋值为base64url编码的DNS请求内容

开发者案例（附python和golang代码示例）

例如用GET方式发起DoH查询，先将DNS二进制报文经过base64url的处理得到dns参数，生成完整的DoH HTTP URL，再发起HTTP请求，获得DNS应答报文。

下面是一段python的代码，演示开发者如何使用DoH查询域名alibaba.com。

```
import dns.message
import requests
import base64
import json

doh_url = "https://dns.alidns.com/dns-query"
domain = "alibaba.com"
rr = "A"
result = []

message = dns.message.make_query(domain, rr)
dns_req = base64.b64encode(message.to_wire()).decode("UTF8").rstrip("=")
r = requests.get(doh_url + "?dns=" + dns_req,
                 headers={"Content-type": "application/dns-message"})
for answer in dns.message.from_wire(r.content).answer:
    dns = answer.to_text().split()
    result.append({"Query": dns[0], "TTL": dns[1], "RR": dns[3], "Answer": dns[4]})
    print(json.dumps(result))
```

运行结果如下：

```
$ python test_doh.py
[{"Answer": "106.11.223.101", "Query": "alibaba.com.", "RR": "A", "TTL": "133"}]
```

下面是一段Golang的代码，演示开发者如何使用DoH查询域名alibaba.com。

```
package main

import (
        "encoding/base64"
        "fmt"
        "github.com/miekg/dns"
        "io/ioutil"
        "net/http"
        "os"
)

func main() {
       query := dns.Msg{}
       query.SetQuestion("www.taobao.com.", dns.TypeA)
       msg, _ := query.Pack()
       b64 := base64.RawURLEncoding.EncodeToString(msg)
       resp, err := http.Get("https://dns.alidns.com/dns-query?dns=" + b64)
       if err != nil {
            fmt.Printf("Send query error, err:%v\n", err)
            os.Exit(1)
       }
       defer resp.Body.Close()
       bodyBytes, _ := ioutil.ReadAll(resp.Body)
       response := dns.Msg{}
       response.Unpack(bodyBytes)
       fmt.Printf("Dns answer is :%v\n", response.String())
}
```

运行结果如下：

```
Sent Get query https://dns.alidns.com/dns-query?dns=xzEBAAABAAAAAAAAA3d3dwZ0YW9iYW8DY29tAAA
BAAE
Dns answer is :;; opcode: QUERY, status: NOERROR, id: 50993
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;www.taobao.com.        IN       A

;; ANSWER SECTION:
www.taobao.com. 15      IN      CNAME   www.taobao.com.danuoyi.tbcache.com.
www.taobao.com.danuoyi.tbcache.com.     15      IN      A       221.229.203.213
www.taobao.com.danuoyi.tbcache.com.     15      IN      A       221.229.203.214

;; ADDITIONAL SECTION:

;; OPT PSEUDOSECTION:
; EDNS: version 0; flags: ; udp: 4096
```

如果您需要接入免费版，不需要带user_id的方式，接入地址格式如下：

- https://dns.alidns.com/dns-query
- https://alidns_ip/dns-query

Firefox的免费接入方式请参考

3.DoH JSON API

DoH JSON API的URL 接口 （提供TLS和非TLS API）

https://dns.alidns.com/resolve?

https://alidns_ip/resolve?

http://dns.alidns.com/resolve?

http://alidns_ip/resolve?

> **注意**
>
> 其中alidns_ip是dns.alidns.com 的A记录，可以是以下两个地址之一： 223.5.5.5，223.6.6.6。

请求方式： GET

请求参数：

参数	类型	描述	实例	使用方法和默认值
name	string	请求域名	name=www.taobao.com.	必选，无默认值
type	integer	请求类型	type=1	可选，1
edns_client_subnet	IP	ECS IP	edns_client_subnet=1.2.3.4/24	DNS代理使用，普通客户端不适用
short	boolean	是否开启简洁模式	short=true 或者 short=1	可选，默认关闭
uid	string	用户ID，即控制台上的AccountID	uid=9999	可选
did	string	设备ID	did=afck0100	可选

关于edns_client_subnet参数：

edns_client_subnet是为了支持DNS ECS功能（RFC7871），将用户的子网信息传递给权威DNS，做更精确的DNS解析和流量调度。其中掩码越长地址信息越精确，掩码越短用户隐私效果越好。建议使用“/24”掩码长度。

注：该参数是特地为DNS代理（proxy）使用DoH JSON API场景设计，即用户发送DNS查询给DNS代理，DNS代理通过该参数携带用户的子网信息传递给阿里公共DNS，最后传递到权威DNS服务器。

例如edns_client_subnet=1.2.3.4/24，权威服务器会收到基于1.2.3.0/24地址前缀信息来帮助用户选择DNS链路。

关于type参数支持类型：

记录类型	ID	意义	示例（以 taobao.com，www.taobao.com 为例）
A	1	IPv4地址	101.37.183.171
NS	2	NS记录	ns1.taobao.com.
CNAME	5	域名 CNAME 记录	www.taobao.com.danuoyi.tbcache.com.
SOA	6	ZONE 的 SOA 记录	ns4.taobao.com. hostmaster.alibabadns.com. 2018011109 3600 1200 3600 360
TXT	16	TXT 记录	"v=spf1 include:spf1.staff.mail.aliyun.com -all"
AAAA	28	IPv6 地址	240e:e1:f300:1:3::3fa

请求示例：

http://dns.alidns.com/resolve?name=www.taobao.com.&type=1

返回示例：

```
{
    "Status":0,
    "TC":false,
    "RD":true,
    "RA":true,
    "AD":false,
    "CD":false,
    "Question":{        // 请求段
        "name":"www.taobao.com.",
        "type":1
    },
    "Answer":[          // 应答段
        {
            "name":"www.taobao.com.",
            "TTL":45,
            "type":5,
            "data":"www.taobao.com.danuoyi.tbcache.com."
        },
        {
            "name":"www.taobao.com.danuoyi.tbcache.com.",
            "TTL":45,
            "type":1,
            "data":"47.246.24.234"
        },
        {
            "name":"www.taobao.com.danuoyi.tbcache.com.",
            "TTL":45,
            "type":1,
            "data":"47.246.24.233"
        }
    ]
    //"Authority" 权威段，如果有数据与Answer字段一致
    //"Additional" 附加段，如果有数据与Answer字段一致
    // 可选 "edns_client_subnet":"1.2.3.4/24"
}
```

返回值说明：

字段名	描述	示例
Status	DNS 报文头的 rcode	0: noerror 2: servfail 3: nxdomain
TC	DNS 报文头的 TC，标识是否可截断	通常为 false

字段名	描述	示例
RD	DNS 报文头的 RD，表示是否期望递归	通常为 true
RA	DNS 报文头的 RA，表示是否为可用递归	通常为 true
AD/CD	对应的 DNS 报文头的标识	通常为 false
Question	DNS 请求字段	
Answer	DNS 应答字段	
name	域名，Question 和 Answer 都包含	www.taobao.com.
type	请求类型，参考上文中“**关于type参数支持类型：**”	
TTL	应答值在服务器中的最大缓存时间，单位为秒	3600
data	应答结果，与 type 相关	

short 模式应答示例：

short模式仅提取，请求中对应 type 的应答列表。

以www.taobao.com为例，当请求为 A 记录时，请求 url 为：

```
http://223.5.5.5/resolve?name=www.taobao.com&type=A&short=1
```

应答如下：

```
["221.229.203.214","61.155.221.227","221.229.203.213"]
```

当请求为 CNAME 记录时，url 为：

```
http://223.5.5.5/resolve?name=www.taobao.com&type=CNAME&short=1
```

应答如下：

```
["www.taobao.com.danuoyi.tbcache.com."]
```

失败应答：

请求失败时，HTTP响应的状态码为4xx/5xx，同时返回具体的错误码，用于调试或者障碍上报，结果用 json 表示。

失败应答实例：

```
{
    "code":"UrlParameterError"
}
```

错误码列表如下：

错误码	HTTP状态码	描述
UrlParameterError	400	请求参数格式错误
NoPermission	401	用户鉴权失败
UrlPathError	404	URL错误
NoResponse	500	请求超时无应答

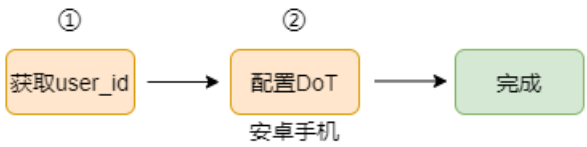
注：用户可以在客户端应用或APP中调用DoH JSON API解析DNS 。

4.DNS over TLS （DoT）

本章节介绍DoT的接入过程。

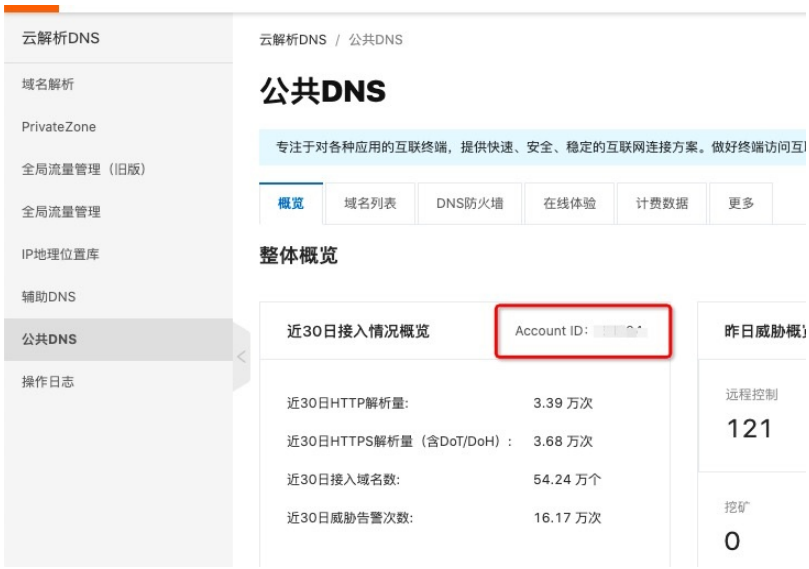
阿里公共DNS通过RFC 7858指定的经过TLS加密的TCP连接提供DNS解析。

基本流程是：



获取user_id的方式

在阿里云控制台开通公共DNS服务之后，即可在概览页面找到 Account ID。如下：



安卓手机接入示例

安卓手机的用户，可以在手机设置界面设置阿里公共DNS的域名和地址来获取DNS的DoT安全传输服务。以某品牌手机为例，如下：

1、假设 Account ID为9999，则配置如下

第一步：

进入设置界面，点击【无线和网络】

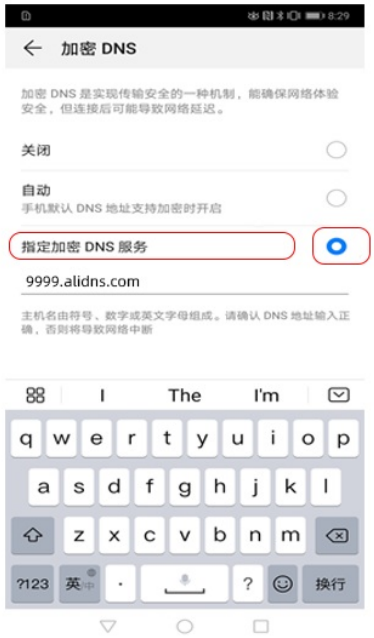
第二步：

点击【加密DNS】

第三步：

选中【指定加密DNS服务】，

并输入阿里云公共DNS服务接口地址。



注：以上介绍的是公共DNS售卖版的接入方式，如果您仅需要接入公共DNS免费版，请参考免费版接入。

5.Android SDK开发指南

本文档介绍了阿里云公共DNS Android SDK的接入和开发方式。

1.概述

阿里云公共DNS SDK是阿里云面向广大移动开发者提供DNS域名解析服务的开发工具包。

开发者利用本SDK，可以在自己的Android APP中轻松接入阿里云公共DNS，解决域名解析异常的问题，低成本实现域名解析精准调度。您可以参考Demo示例工程源码了解如何使用本SDK。

SDK当前版本封装了阿里公共DNS的DoH JSON API，提供Java函数接口给Android APP进行域名解析，并且提供了基于TTL和LRU策略的高效域名缓存功能。在公共DNS原有功能的基础上，SDK还可以为用户带来以下优势：

- **简单易用：**

  用户仅需集成我们提供的SDK，便可接入阿里云公共DNS业务。接入方法简单易用，为用户提供更为轻松便捷的解析服务。

- **零延迟：**

  SDK内部实现了LRU的缓存机制，将每次域名解析后的IP缓存到本地；并且主动更新TTL过期缓存，保证缓存及时有效，从而帮助用户达到域名解析零延迟的效果。

2.如何使用SDK

2.1 jar集成

1. 您需先在控制台注册自己的应用，获取应用的唯一标识Account ID
2. 然后通过控制台的链接获取阿里公共DNS SDK
3. 获取到的SDK的alidns_android_sdk.jar集成到自己工程项目libs目录中即可轻松使用

2.2 gradle集成maven

在 build.gradle 文件中加入以下代码：

```
allprojects {
  repositories {
    maven {
      url 'https://maven.aliyun.com/repository/public/'
    }
    mavenLocal()
    mavenCentral()
  }
}
```

加入你要引用的文件信息：

```
dependencies {
     implementation 'com.alibaba.pdns:alidns-android-sdk:2.1.0'
     implementation 'com.google.code.gson:gson:2.8.5'
}
```

2.3 应用程序初始化

```
public class DnsCacheApplication extends Application{

    private String accountID="***"; //设置您在控制台接入sdk的accountId
    private static final String TAOBAO_HOST_NAME = "www.taobao.com";//请替换为您要预加载的域名
    private static final String ALIYUN_HOST_NAME = "www.aliyun.com"; //请替换为您要预加载的域
名
    private static final int CACHE_MAX_NUMBER=100; //设置缓存域名的最大个数，默认为100
    private static final int MAX_NEGATIVE_CACHE = 30; //设置否定缓存的最大TTL时间，默认为30秒
    private static final int MAX_TTL_CACHE = 1 * 60 * 60; //设置缓存的最大TTL时间，默认为3600秒

    @Override
    public void onCreate() {
       super.onCreate();
       DNSResolver.Init(this, accountID);     //设置控制台接入sdk的accountId
       DNSResolver.setAccessKeySecret(""); //设置控制台接入sdk的accessKeySecret
       DNSResolver.setAccessKeyId("");       //设置控制台接入sdk的accessKeyId
       DNSResolver.setEnableShort(false); //设置是否开启short模式，默认不开启
       DNSResolver.setEnableIPv6(false);    //设置是否开启IPV6的访问模式，默认不开启
       DNSResolver.setEnableCache(true); //设置是否开启使用缓存，默认开启
       DNSResolver.setEnableSpeedTest(false); //设置是否开启IP测速，默认不开启
       DNSResolver.setEnableSchedulePrefetch(true); //设置是否开启定时主动更新过期缓存,默认为开
启
       DNSResolver.setMaxTtlCache(MAX_TTL_CACHE); //设置缓存的最大TTL时间，默认为3600秒
       DNSResolver.setSpeedPort(DNSResolver.PORT_80); //设置ip测量socket检测的端口号，默认80端
口
       DNSResolver.setMaxNegativeCache(MAX_NEGATIVE_CACHE); //设置否定缓存的最大TTL时间，默认为
30秒
       DNSResolver.setSchemaType(DNSResolver.HTTP); //设置访问http还是https模式
       DNSResolver.getInstance().setMaxCacheSize(CACHE_MAX_NUMBER); //设置缓存域名的最大个数，
默认为100
       DNSResolver.getInstance().preLoadDomains(new String[]{TAOBAO_HOST_NAME,ALIYUN_HOST_N
AME}); //设置域名预先解析,将预加载域名替换为您希望使用阿里DNS解析的域名
    }
}
```

接入阿里公共DNS SDK时建议在Application子类里集成。

> **说明**
>
> DNSResolver为阿里公共DNS SDK的核心类，其内部封装了阿里公共DNS提供的DoH JSON API，将用户的目标域名解析成为对应的IP。

另外，Android工程在使用SDK时，需要保证提供以下访问权限的配置：

```
<!--需要配置的权限-->
   <uses-permission android:name="android.permission.INTERNET"/>
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
   <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

注意事项：

1). Android 9.0 发送http网络请求会报cleartext HTTP traffic no permitted异常

- 原因：从Android9.0（API 28）开始，Android默认将禁止明文访问网络只允许使用https url访问。
- 解决方案

在应用的AndroidManifest.xml文件的<application>中添加代码：

android:usesCleartextTraffic="true"

```
<application
        android:name=".DnsCacheApplication"
        android:icon="@mipmap/ic_launcher"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="true">
```

2). Android9.0会报异常：Didn't find class BasicHttpParams。

- 原因：Apache Http客户端弃用 。

因为早在Android 6.0中，谷歌取消了对Apache Http客户端的支持。从Android 9.0开始，org.apache.http.legacy将从bootclasspath中删除。

该修改对于大多数 taskVersion < 9.0的应用没有影响，对所有taskVersion > 9.0的应用，如果继续使用Apache Http接口或者引用的lib包中用到该接口时，都会出现Apache Http接口找不到的异常。

- 解决方案

在应用的AndroidManifest.xml文件的<application>中添加：

```
<uses-library android:name="org.apache.http.legacy" android:required="false"/>
```

3). 本SDK从2.0开始支持设置鉴权认证功能，来保障用户身份安全，不被第三方未授权者盗用。

如果用户不设置以下参数则不会开启鉴权功能，若要开启鉴权功能，请参考产品鉴权文档在控制台上创建AccessKey ID和AccessKey Secret，可以通过以下2种方式添加配置：

- 在应用的AndroidManifest.xml文件中添加以下配置

```
<meta-data
        android:name="com.alibaba.pdns.ACCESS_KEY_ID"
        android:value="替换成您在阿里云控制台注册app时生成的AccessKey ID" />

<meta-data
        android:name="com.alibaba.pdns.ACCESS_KEY_SECRET"
        android:value="替换成您在阿里云控制台注册app时生成的AccessKey Secret" />
```

- 在应用接入SDK初始化时设置参数：

```
DNSResolver.setAccessKeySecret("");    //设置控制台接入sdk的accessKeySecret
DNSResolver.setAccessKeyId("");        //设置控制台接入sdk的accessKeyId
```

> **说明**
>
> 以上两种配置鉴权认证方式，开发者可根据实际应用场景任选其一。当配置了其中的一种则无需再重复配置另外一种方式。

4). 为了更好的使用本SDK，建议您在编译APP时将JDK的版本设置为1.8。

2.4.1 服务初始化

阿里公共DNS SDK是通过DNSResolver类封装了阿里公共DNS服务请求和本地缓存实现的。用户通过DNSResolver.Init(this,accountID)初始化，即可统一接入阿里公共DNS服务。（accountID为用户在控制台注册时，server端自动生成用户唯一标识）

2.4.2 设置预解析域名

在您初始化程序时，可以选择性地预先向阿里公共DNS SDK中注册您后续可能会使用到的域名，以便SDK提前解析，减少后续解析域名时请求的时延。调用以下方法设置预解析域名：

- 指定预加载IPV6或IPV4域名解析

```
//设置域名预先解析,将预加载域名替换为您希望使用阿里DNS解析的域名
DNSResolver.getInstance().preLoadDomains(DNSResolver.QTYPE_IPV4,new String[]{...})

DNSResolver.QTYPE_IPV4   预取域名解析的IPV4记录类型
DNSResolver.QTYPE_IPV6   预取域名解析的IPV6记录类型
```

- 根据当前网络自动匹配预加载IPV6或IPV4域名解析

```
DNSResolver.getInstance().preLoadDomains(domains)
```

> **注意**
>
> 预解析接口设置的同时会实时触发异步网络请求，应该在代码逻辑上确保调用预解析接口时，已经进行了必备的初始化设置。

2.4.3 设置是否使用服务端IPV6地址

阿里公共DNS服务支持IPV4、IPV6双栈访问，通过设置DNSResolver.setEnableIPv6(boolean enable)的方法选择使用那种接入协议。当设置为ture表示使用IPV6的IP访问服务端接口，false表示使用IPV4的IP访问服务端接口。如不显式设置，默认为使用IPV4地址访问。另外当设置为IPV6时访问阿里公共DNS服务不通时则自动切换为IPV4，并且支持重试3次策略。

2.4.4 设置是否开启Short模式

阿里公共DNS的DoH JSON API返回数据类型分为全量JSON和简要IP数组格式，可以通过调用DNSResolver.setEnableShort (boolean enable)来开启或关闭short模式。如不显式设置，默认为关闭short模式。

如下方式：

```
DNSResolver.setEnableShort (true); //默认值是false,该参数用户可以不需设置，不设置则默认为false。
```

> **注意**
>
> short模式为SDK调用阿里公共DNS服务返回的比较简单IP数组，可以减少回复的数据量，适用于对网络流量敏感的场景。

2.4.5 设置缓存的最大个数

```
DNSResolver.getInstance().setMaxCacheSize(CACHE_MAX_NUMBER); //设置缓存的最大个数，默认为100
```

用户可以自定义最大count的值。

2.4.6设置缓存的最大TTL时间

```
DNSResolver.setMaxTtlCache(MAX_TTL_CACHE);//设置缓存的最大TTL时间，默认为3600秒
```

2.4.7 设置访问服务端协议

```
DNSResolver.setSchemaType(DNSResolver.HTTP); 默认访问为http的模式。
```

DNSResolver.HTTP，以HTTP协议访问服务端接口

DNSResolver.HTTPS，以HTTPS协议访问服务端接口

2.4.8 设置是否开启使用缓存

```
DNSResolver.setEnableCache(true); //默认开启使用缓存
```

用户可以设置是否开启使用缓存功能（ture为开启缓存功能，false为关闭缓存功能）

2.4.9 设置是否开启IP测速

```
DNSResolver.setEnableSpeedTest(false); //默认不开启IP测速
```

用户可以设置是否开启IP测速（ture为开启IP测速功能，false为关闭IP测速功能）

2.4.10 设置IP测速socket监测的端口号

```
DNSResolver.setSpeedPort(DNSResolver.PORT_80)
```

用户可以设置IP测速基于socket监测的端口号，默认为80

2.4.11 设置是否开启依据ISP网络区分域名缓存

```
DNSResolver.setIspEnable(true);//是否开启依据ISP网络区分域名缓存
```

设置是否开启依据ISP网络区分域名缓存。如果开启，则在不同网络环境下域名缓存数据分别存储互不影响。如果不开启，则不同网络下使用同一份域名缓存数据

2.4.12 设置否定缓存的最大TTL时间

```
DNSResolver.setMaxNegativeCache(MAX_NEGATIVE_CACHE);//设置否定缓存的最大TTL时间，默认为30秒
```

用户可以根据自身的需求设置否定缓存的最大TTL时间

2.4.13 设置鉴权认证AccessKeySecret、AccessKeyId参数

```
DNSResolver.setAccessKeySecret("");    //设置控制台接入sdk的accessKeySecret
DNSResolver.setAccessKeyId("");        //设置控制台接入sdk的accessKeyId
```

2.4.14 设置是否开启定时主动更新过期缓存

本SDK在缓存功能已开启的情况下，可设置开启定时主动更新过期缓存功能。如果该功开启后，SDK会每分钟主动更新过期缓存一次，可以有效保障用户缓存数据得到及时更新。但可能会带来域名解析次数和客户端网络流量消耗的增加。

```
DNSResolver.setEnableSchedulePrefetch(true); // 设置是否开启定时主动更新过期缓存,默认为开启
```

3.反混淆配置

```
-keep class com.alibaba.pdns.** {*;}
```

4.服务API

```
/**
  * 通过自动感知当前网络环境（ipv4-only、ipv6-only、ipv4和ipv6双栈）获取该域名的解析数据
  *
  * @param host 用户所要解析的域名
  * @return 返回根据当前网络环境下的最优IP数组
  */
public String[] getIpsByHost(String host)


 /**
  * 通过自动感知当前网络环境（ipv4-only、ipv6-only、ipv4和ipv6双栈）预加载域名解析
  *
  * @param domains 预加载解析的域名
  */
  public void preLoadDomains(final String[] domains)


/**
  * 通过自动感知当前网络环境（ipv4-only、ipv6-only、ipv4和ipv6双栈）从缓存中获取该域名解析后的IP数组
  *
  * @param host 查询的host 例如（www.taobao.com)
  * @param isAllowExp 是否允许返回已过期域名的解析数据
  * @return 获取缓存中对应host解析后的IP数组
  */
   public String[] getIpsByHostFromCache(String host, boolean isAllowExp)


/**
 * 获取url对应IPv4记录的DomainInfo对象数组
 *
 * @param url例如(http://www.taobao.com)
```

```
  * @param url例如(http://www.taobao.com)
  * @return目标url对应IPV4类型的DomainInfo对象数组
  */
    public DomainInfo[] getIPsV4DInfoByUrl(String url)
```

注意：DomainInfo对象中的url为替换host为IP拼接后的url,用户不需再手动url的替换拼接。

```
/**
  * 获取url对应IPv6记录的DomainInfo对象数组
  *
  * @param url 例如（http://m.taobao.com)
  * @return 目标url对应IPV6类型的DomainInfo对象数组
  */
   public DomainInfo[] getIPsV6DInfoByUrl(String url)


 /**
   * 获取url对应IPV4记录的DomainInfo对象
   *
   * @param url 例如(http://m.taobao.com)
   * @return 目标url对应IPV4类型的DomainInfo对象集合中随机的一个
   */
    public DomainInfo getIPV4DInfoByUrl(String url)


/**
   * 获取url对应IPV6记录的DomainInfo对象
   *
   * @param url 例如(http://www.taobao.com)
   * @return 目标url对应IPV6类型的DomainInfo对象集合中随机的一个
   */
   public DomainInfo getIPV6DInfoByUrl(String url)
```

说明：返回的domainInfo对象，主要是封装了如下属性

```
 /**
  * id访问域名自增的id编号
   */
   public String id = null;

  /**
   * 可以直接使用的url 已经替换了host为ip后的url
   */
    public String url = null;

   /**
   * 需要设置到http header里面的目标服务名称
   */
   public String host = "";

  /**
   * 返回的内容体
   */
  public String data = null;
```

```
    /**
     * 开始请求的时间
     */
    public String startTime = null;

    /**
     * 请求结束的时间，如果请求超时，该值为null
     */
    public String stopTime = null;

    /**
    * server返回的状态值200 \ 404 \ 500等
    */
    public String code = null;

   /**
     * 获取hostName对应IPV4记录数组
     * @param hostName例如(www.taobao.com)
     * @return 返回目标hostName对应的IPV4地址的数组
     */
   public  String[] getIPsV4ByHost(String hostName)

/**
     * 获取hostName对应IPV6记录数组
     * @param hostName例如(www.taobao.com)
     * @return 返回目标hostName对应的IPV6地址的数组
     */
   public String[] getIPsV6ByHost(String hostName)

/**
     * 获取hostName对应IPV4记录
     * @param hostName例如(www.taobao.com)
     * @return 返回目标hostName对应的IPV4地址集合中的一个随机IPV4地址,当开启测速返回一个最优IPV4地址
     */
   public String getIPV4ByHost(String hostName)

/**
     * 获取hostName对应IPV6记录
     * @param hostName例如(www.taobao.com)
     * @return 返回目标hostName对应的IPV6地址集合中的一个随机IPV6地址,当开启测速返回一个最优IPV6地址
     */
  public String getIPV6ByHost(String hostName)

    /**
      *  从缓存中获取该域名解析后的IPV4记录类型的IP数组
      *  如果缓存没有，该方法是先直接返回null，然后同时进行异步查询，最后将查询到的数据存入缓存中
      *
      * @param host 查询的host 例如(www.taobao.com)
      * @param isAllowExp 是否允许返回已过期域名的解析数据
      * @return 获取缓存中对应host解析后的IPV4记录类型的IP数组
      */
     private String[] getIpv4ByHostFromCache(String host , boolean isAllowExp)

     /**
```

```
     *  从缓存中获取该域名解析后的IPV6记录类型的IP数组
     *  如果缓存没有，该方法是先直接返回null，然后同时进行异步查询，最后将查询到的数据存入缓存中
     *
     * @param host 查询的host 例如(www.taobao.com)
     * @param isAllowExp 是否允许返回已过期域名的解析数据
     * @return 获取缓存中对应host解析后的IPV6记录类型的IP数组
     */
    private String[] getIpv6ByHostFromCache(String host , boolean isAllowExp)

    /**
     * 预加载域名解析
     *
     * @param qType 预加载解析IPV4还是IPV6
     * @param domains 预加载解析的域名
     */
    public void preLoadDomains(String qType, final String[] domains)

    /**
     * 获取请求阿里公共DNS成功失败统计信息
     *
     * @return 所有的域名解析统计信息Json数组字符串
     */
    public String getRequestReportInfo()
```

5.SDK API使用示例

URl：传进来的访问地址，例如：http://www.taobao.com。

```
String hostname = "www.taobao.com";
String url = "http://www.taobao.com";
```

5.1 获取当前网络环境下的最优IP数据：

```
String[] ip = DNSResolver.getInstance().getIpsByHost(hostname); //获取当前网络下的最优域名解析IP
```

5.2 根据当前网络环境预加载域名解析：

```
DNSResolver.getInstance().preLoadDomains(domains) //设置域名预先解析,将预加载域名替换为您希望使用阿里DNS解析的域名
```

5.3 根据当前网络环境读取缓存中的域名解析数据：

```
String[] ip = DNSResolver.getInstance().getIpsByHostFromCache(hostname,true);//获取当前网络环境缓存中的域名解析数据
```

5.4 获得IPv4地址：

```
String IPV4 = DNSResolver.getInstance().getIPV4ByHost(hostname); //获取IPV4的域名解析
```

5.5 获得IPv6地址：

```
String IPV6 =  DNSResolver.getInstance().getIPV6ByHost(hostname) //获取IPV6的域名解析
```

5.6 从缓存中获取ipv4解析地址

```
String[] IPV4 =  DNSResolver.getInstance().getIpv4ByHostFromCache(hostname , true) //获取缓存里IPV4的域名解析
```

5.7 从缓存中获取ipv6解析地址

```
String[] IPV6 =  DNSResolver.getInstance().getIpv6ByHostFromCache(hostname , true) //获取缓存里IPV6的域名解析
```

5.8 获得url对应的DomainInfo对象：

```
DomainInfo dinfo = DNSResolver.getInstance().getIPV4DInfoByUrl(url); //获取替换的URL
```

5.9 获取请求阿里DNS成功失败统计信息

```
String reportInfo = DNSResolver.getInstance().getRequestReportInfo(); //获取成功失败统计信息
```

所有的域名解析的统计信息Json字符串数组中的字段含义描述如下：

```
[
    {
      avgRtt = "1";                          // 域名平均解析时间ms
      hitDnsCacheCount = 0;                  // 命中缓存次数
      domainName = "www.taobao.com";         // 解析的域名
      degradeLocalDnsCount= 0;               // 降级到LocalDNS的次数
      localDnsResolveErrCount = 0;           // LocalDNS解析失败次数
      maxRtt = "60";                         // 域名最大解析时间ms
      permissionErrCount= 0;                 // 用户鉴权失败次数
      reqServerErrCount= 0;                  // 请求超时无应答次数
      recursiveReqCount= 1;                  // 递归查询的次数
      isp = 中国移动;                        // 运营商名字
      resolveSuccessCount= 1;                // 解析成功次数
      reqTimeoutCount = 0;                   // 网络超时错误次数
      queryType= 28;                         // ip类型1代表ipv4、28代表ipv6
      reqParameterErrCount = 0;              // 请求参数格式错误次数
      reqPathErrCount= 0;                    // URL错误次数
    }
      ......
]
```

> **注意**
>
> 请求阿里公共DNS域名解析统计数据是以**网络环境+域名+请求类型**为统计维度

6.示例：

```
public class MainActivity extends AppCompatActivity {
   private Button button;
   private TextView tvInfo;
   private TextView tvResult;
   private String hostUrl = "http://www.taobao.com"; //请替换为您需要解析hostUrl
   private String hostName = "www.taobao.com"; //请替换为您需要解析hostName
   private static final String TAG = "PDnsDemo";
   private static ExecutorService pool = Executors.newSingleThreadExecutor();
   private static final String PDNS_RESULT = "pdns_result";
   private static final int SHOW_CONSOLE_TEXT = 10000;
   private Handler mHandler;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.demo_activity_main);
       init();
       initHandler();
   }
```

```
 private void init() {
       tvInfo = findViewById(R.id.tv_respons_info);
       tvResult = findViewById(R.id.tv_respons);
       button = findViewById(R.id.btn_onclik);
       button.setOnClickListener(new View.OnClickListener() {
           public void onClick(View view) {
               new Thread(new Runnable() {
                       @Override
                       public void  run() {
                       //调用阿里公共sdk里的getIPV4ByHost()方法获得目标域名解析后的IP
                       String ipv4 = DNSResolver.getInstance().getIPV4ByHost(hostName);
                       tvInfo.setText("您解析域名的IP是："+ ipv4);
                       //调用阿里公共sdk里getIPV4DInfoByUr获取目标域名解析后的domainInfo对象中的U
RL，该URL为已替换原来url中host为IP的url
                       DomainInfo dinfo = DNSResolver.getInstance().getIPV4DInfoByUrl(hostUr
l);
                       if (dinfo != null) {
                            showResponse(dinfo);
                        }
                   }
               }).start();
           }
       });
   }
```

```
private void initHandler() {
    mHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            switch (msg.what)  {
                case SHOW_CONSOLE_TEXT:
                    tvResult.setText(msg.getData().getString(PDNS_RESULT) + "\n");
                    break;
            }
        }
    };
}
```

```
private void showResponse(final DomainInfo dinfo) {
             //发送网络请求
            String requestUrl = dinfo.url;
            HttpURLConnection conn = null;
            try {
                URL url = new URL(requestUrl);
                conn = (HttpURLConnection) url.openConnection();
                //使用IP的方式进行访问时，需要设置HTTP请求头的HOST字段为原来的域名
                conn.setRequestProperty("Host", url.getHost());//设置HTTP请求头HOST字段
                DataInputStream dis = new DataInputStream(conn.getInputStream());
                int len;
                byte[] buff = new byte[4096];
                StringBuilder response = new StringBuilder();
                while ((len = dis.read(buff)) != -1) {
                    response.append(new String(buff, 0, len));
                }
                Log.d(TAG, "Response: " + response.toString());
                dis.close();
                sendMessage(response.toString());
            } catch (IOException e) {
                e.printStackTrace();
            }finally {
                if (conn != null) {
                    conn.disconnect();
                }
            }
        }
```

```
    private void sendMessage(String message) {
        if (mHandler != null) {
                Message msg = mHandler.obtainMessage();
                Bundle bundle = new Bundle();
                bundle.putString(PDNS_RESULT, message);
                msg.setData(bundle);
                msg.what = SHOW_CONSOLE_TEXT;
                mHandler.sendMessage(msg);
        }
    }
}

public class DnsCacheApplication extends Application {
    private String accountID = "****"; //可以更换成您的accountID
    private static final String TAOBAO_HOST_NAME = "www.taobao.com";//请替换为您要预加载的域名
    private static final String ALIYUN_HOST_NAME = "www.aliyun.com";//请替换为您要预加载的域名
    private static final int CACHE_MAX_NUMBER = 100; //缓存域名的最大个数,默认为100
    private static final int MAX_NEGATIVE_CACHE = 30; //设置否定缓存的最大TTL时间,默认为30秒
    private static final int MAX_TTL_CACHE = 1 * 60 * 60; //设置缓存的最大TTL时间,默认为3600秒

    @Override
    public void onCreate() {
         super.onCreate();
         DNSResolver.Init(this, accountID);//设置控制台接入sdk的accountId
         DNSResolver.setAccessKeySecret("");//设置控制台接入sdk的accessKeySecret
         DNSResolver.setAccessKeyId("");    //设置控制台接入sdk的accessKeyId
         DNSResolver.setEnableShort(false);//设置是否开启short模式
         DNSResolver.setEnableIPv6(false);//设置是否开启IPV6的访问模式
         DNSResolver.setEnableCache(true);//设置是否开启使用缓存
         DNSResolver.setEnableSpeedTest(false);/设置是否开启IP测速
         DNSResolver.setEnableSchedulePrefetch(true); // 设置是否开启定时主动更新过期缓存,默认为
开启
         DNSResolver.setIspEnable(true);//设置是否开启依据ISP网络区分域名缓存
         DNSResolver.setMaxTtlCache(MAX_TTL_CACHE); //设置缓存的最大TTL时间，默认为3600秒
         DNSResolver.setMaxNegativeCache(MAX_NEGATIVE_CACHE);//设置否定缓存的最大TTL时间，默认为
30秒
         DNSResolver.setSchemaType(DNSResolver.HTTP);//设置访问http还是https模式
         DNSResolver.setSpeedPort(DNSResolver.PORT_80);//设置ip测量socket检测的端口号，默认80端
口
         DNSResolver.getInstance().setMaxCacheSize(CACHE_MAX_NUMBER);//设置缓存域名的最大个数，
默认为100
         DNSResolver.getInstance().preLoadDomains(new String[]{TAOBAO_HOST_NAME,ALIYUN_HOST_
NAME });//设置域名预先解析,预加载域名替换为您希望使用阿里DNS解析的域名
    }
}
```

注意事项

1. 通过阿里公共DNS获得域名的IP地址后，客户端可以使用这个IP发送业务请求，HTTP请求头的Host字段需改为原来的域名。
2. 为帮助用户更快的使用阿里公共DNS SDK，我们为读者提供Demo程序，读者可以

   下载到本地作为参考。

请点击这里，下载Demo程序。

6.iOS SDK开发指南

本文档介绍了阿里云公共DNS iOS SDK的接入和开发方式。

1.概述

阿里云公共DNS SDK是阿里云面向广大移动开发者提供DNS域名解析服务的开发工具包。

开发者利用本SDK，可以在自己的iOS APP中轻松接入阿里云公共DNS，解决域名解析异常的问题，低成本实现域名解析精准调度。您可以参考Demo示例工程源码了解如何使用本SDK。

iOS 14 开始系统原生支持两种标准规范的 Encrypted DNS, 分别是 DNS over TLS 与 DNS over HTTPS，您可以参考iOS14原生加密DNS方案了解如何设置阿里云公共DNS为加密DNS默认解析器。

SDK当前版本封装了阿里公共DNS的DoH JSON API，提供接口函数给iOS APP进行域名解析，并且提供了基于TTL和LRU策略的高效域名缓存功能。在公共DNS原有功能的基础上，SDK还可以为用户带来以下优势：

- **简单易用：**

  用户仅需集成我们提供的SDK，便可接入阿里云公共DNS业务。接入方法简单易用，为用户提供更为轻松便捷的解析服务。

- **零延迟：**

  SDK内部实现了LRU的缓存机制，将每次域名解析后的IP缓存到本地；并且主动更新TTL过期缓存，保证缓存及时有效，从而帮助用户达到域名解析零延迟的效果。

2.SDK集成

2.1 SDK集成

1. 您需先在控制台注册自己的应用，获取应用的唯一标识Account ID。
2. 通过控制台的链接获取阿里公共DNS iOS SDK。
3. 获取到的SDK的 `pdns-sdk-ios.framework` 后，手动集成到自己工程中。
4. 引入系统库：
   - Foundation.framework
   - SystemConfiguration.framework
   - CoreFoundation.framework
   - CoreTelephony.framework
5. 工程Build Settings中Other Linker Flags 加入 `-ObjC`
6. 在 `application:didFinishLaunchingWithOptions:` 初始化SDK。

```
DNSResolver *resolver = [DNSResolver share];
resolver.accountId = @"您在控制台注册应用时分配的Account ID";
```

2.2 自动集成（Cocoapods）

1.Podfile中指定仓库位置：（Master仓库不要遗漏）

```
source 'https://github.com/CocoaPods/Specs.git'
source 'https://github.com/aliyun/aliyun-specs.git'
```

2.为工程target添加依赖：

```
pod 'AlicloudPDNS'
```

3.API介绍

3.1 accountId

必传参数，您在控制台注册自己的应用后，控制台会为此应用生成唯一标识Account ID。

3.2 设置是否使用服务端IPV6地址

阿里云公共DNS服务支持IPv4、IPv6双栈访问。SDK默认使用IPv4地址访问DNS服务器进行解析。

如果需要通过IPv6地址访问DNS服务器进行解析（当前网络需要支持IPv6），则需要通过以下代码设置：

```
[DNSResolver share].ipv6Enable = YES;
```

3.3 short模式

阿里公共DNS的DoH JSON API返回数据分为全量JSON和简要IP数组格式。SDK默认为全量JSON格式。

若要设置为简要IP数组格式，则需要通过以下代码设置：

```
[DNSResolver share].shortEnable = YES;
```

3.4 scheme

在访问DNS服务器进行解析时，是通过HTTP协议解析，还是通过HTTPS协议解析，可通过 `scheme` 属性进行设置。

SDK默认使用http协议进行解析，建议使用http协议进行解析，解析速度更快。若要使用https协议进行解析，则需要如下设置：

```
[DNSResolver share].scheme = DNSResolverSchemeHttps;
```

3.5 设置是否开启缓存

SDK可设置开启缓存功能，如果缓存功能开启，在第一次解析过域名后，后续再解析时， 优先获取缓存中的数据，可大大提高解析速度。

SDK默认开启缓存。若要设置不开启缓存，则需要通过以下代码设置：

```
[DNSResolver share].cacheEnable=NO;
```

3.6 设置是否开启定时更新过期缓存

SDK在缓存功能已开启的情况下，可设置开启定时主动更新过期缓存功能，如果该功能开启，SDK每分钟会自动更新过期缓存，保障用户缓存数据及时更新，但是可能会带来域名解析次数和客户端流量消耗的增多。

该功能默认开启，若要设置关闭，则需要通过以下代码设置：

```
[DNSResolver share].schedulePrefetchEnable=NO;
```

3.7 设置缓存数量

SDK若开启了缓存功能，可以自定义缓存数量（支持范围100～500之间）。

SDK默认的缓存数量为100个域名。若要自定义缓存数量，可通过cacheCountLimit 属性进行设置：

```
[DNSResolver share].cacheCountLimit = 200;
```

3.8 设置是否开启IP测速

SDK可设置开启IP测速，如果IP测速开启，解析结果会优先返回速度最快的IP，数组会按照测速结果由快到慢的顺序排列。

SDK默认没有开启IP测速。若要设置开启IP测速，则需要通过以下代码设置：

```
[DNSResolver share].speedTestEnable=YES;
```

3.9 设置IP测速方式

SDK可设置IP测速的方式，如果IP测速开启，并且设置该参数为0，则使用ICMP探测；如果设置该参数为80、443或其它支持的端口号，则为socket指定端口探测。

SDK默认该参数为0。若要设置socket指定端口探测，则需要通过以下代码设置：

```
[DNSResolver share].speedPort = 80;
```

3.10 设置是否开启依据ISP网络区分域名缓存

SDK可设置是否开启依据ISP网络区分域名缓存。如果开启，则在不同网络环境下域名缓存数据分别存储互不影响。如果不开启，则不同网络下使用同一份域名缓存数据。

SDK默认没有开启依据ISP网络区分域名缓存功能，若要开启该功能，则需要通过以下代码设置：

```
[DNSResolver share].ispEnable = YES;
```

3.11 设置否定缓存最大TTL时间

SDK可设置否定缓存的最大TTL时间，如果设置了该时间，则会限制否定缓存的最大TTL不超过该设置时间。

SDK默认该参数为30s。若要设置否定缓存最大TTL时间，则需要通过以下代码设置：

```
[DNSResolver share].maxNegativeCache = 30;
```

3.12 设置缓存最大TTL时间

SDK可设置缓存的最大TTL时间，如果设置了该时间，则会限制缓存的最大TTL不超过该设置时间。

SDK默认该参数为3600s。若要设置缓存最大TTL时间，则需要通过以下代码设置：

```
[DNSResolver share].maxCacheTTL= 3600;
```

3.13 timeout

`timeout` 属性为域名解析的超时时间。默认超时时间为3s，用户可自定义超时时间，建议设置在2~5s之间。

3.14 设置开启鉴权

SDK在2.0.0版本后可设置鉴权功能来保障用户身份安全，不被第三方未授权者盗用。

如果用户不设置以下参数则不会开启鉴权功能，若要开启鉴权功能，参考产品鉴权文档在控制台创建AccessKey，并在APP中通过如下代码设置：

```
[DNSResolver share].accessKeyId = @"你的AccessKey ID";
[DNSResolver share].accesskeySecret = @"你的Accesskey Secret";
```

3.15 预解析

由于SDK自带缓存功能，在第一次解析完域名后，后续再次解析此域名可大大提高解析速度，所以建议在app启动后，可对app中可能要解析的域名进行预加载。

代码示例：

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionar
y *)launchOptions {
   // Override point for customization after application launch.

   DNSResolver *resolver = [DNSResolver share];
   resolver.accountId = @"你的Accound ID";
   resolver.cacheEnable = YES;
   //对后续可能要解析的域名进行预加载
   [resolver preloadDomains:@[@"域名1", @"域名2", @"域名3"] complete:^{
       //所有域名预加载完成

   }];

   return YES;
}
```

4.服务API

代码示例：

```
/// 预解析域名信息，可在程序启动时调用，加快后续域名解析速度
/// 自动感知网络环境（ipv4-only、ipv6-only、ipv4和ipv6双栈）解析得到适用于当前网络环境的ip
/// @param domainArray  域名数组
/// @param complete     解析完成后回调
- (void)preloadDomains:(NSArray<NSString *> *)domainArray complete:(void(^)(void))complete;

/// 获取域名解析后的ip数组，自动感知网络环境（ipv4-only、ipv6-only、ipv4和ipv6双栈）得到适用于当前网
络环境的ip
/// @param domain           域名
/// @param complete         回调(所有ip地址)
- (void)getIpsDataWithDomain:(NSString *)domain complete:(void(^)(NSArray<NSString *> *data
Array))complete;

/// 自动感知网络环境（ipv4-only、ipv6-only、ipv4和ipv6双栈）直接从缓存中获取适用于当前网络环境的ip数
组，无需等待．  如无缓存，或有缓存但已过期并且enable为NO，则返回 nil
```

```
/// @param domain    域名
/// @param enable    是否允许返回过期ip
- (NSArray<NSString *> *)getIpsByCacheWithDomain:(NSString *)domain andExpiredIPEnabled:(BO
OL)enable;

/// 获取域名解析后的IPv4信息数组
/// @param domain            域名
/// @param complete        回调(所有域名信息)
- (void)getIpv4InfoWithDomain:(NSString *)domain complete:(void(^)(NSArray<DNSDomainInfo *>
*domainInfoArray))complete;

/// 获取域名解析后的IPv6信息数组
/// @param domain            域名
/// @param complete        回调(所有域名信息)
- (void)getIpv6InfoWithDomain:(NSString *)domain complete:(void(^)(NSArray<DNSDomainInfo *>
*domainInfoArray))complete;

/// 获取域名解析后的IPv4信息
/// @param domain            域名
/// @param complete        回调(所有域名信息中随机一个)
- (void)getRandomIpv4InfoWithDomain:(NSString *)domain complete:(void(^)(DNSDomainInfo *dom
ainInfo))complete;

/// 获取域名解析后的IPv6信息
/// @param domain            域名
/// @param complete        回调(所有域名信息中随机一个)
- (void)getRandomIpv6InfoWithDomain:(NSString *)domain complete:(void(^)(DNSDomainInfo *dom
ainInfo))complete;

/// 获取域名解析后的IPv4地址数组
/// @param domain            域名
/// @param complete        回调(所有ip地址)
- (void)getIpv4DataWithDomain:(NSString *)domain complete:(void(^)(NSArray<NSString *> *dat
aArray))complete;

/// 获取域名解析后的IPv6地址数组
/// @param domain            域名
/// @param complete        回调(所有ip地址)
- (void)getIpv6DataWithDomain:(NSString *)domain complete:(void(^)(NSArray<NSString *> *dat
aArray))complete;

/// 获取域名解析后的IPv4地址
/// @param domain            域名
/// @param complete        回调(所有ip地址中随机一个)
- (void)getRandomIpv4DataWithDomain:(NSString *)domain complete:(void(^)(NSString *data))co
mplete;

/// 获取域名解析后的IPv6地址
/// @param domain            域名
/// @param complete        回调(所有ip地址中随机一个)
- (void)getRandomIpv6DataWithDomain:(NSString *)domain complete:(void(^)(NSString *data))co
mplete;

/// 预解析域名IPv4信息，可在程序启动时调用，加快后续域名解析速度
```

```
/// @param domainArray  域名数组
/// @param complete         解析完成后回调
- (void)preloadIpv4Domains:(NSArray<NSString *> *)domainArray complete:(void(^)(void))compl
ete;

/// 预解析域名IPv6信息，可在程序启动时调用，加快后续域名解析速度
/// @param domainArray  域名数组
/// @param complete         解析完成后回调
- (void)preloadIpv6Domains:(NSArray<NSString *> *)domainArray complete:(void(^)(void))compl
ete;

/// 直接从缓存中获取ipv4解析结果，无需等待.  如无缓存，或有缓存但已过期，并且enable为NO，则返回 nil
/// @param domain   域名
/// @param enable   是否允许返回过期ip
- (NSArray<NSString *> *)getIpv4ByCacheWithDomain:(NSString *)domain andExpiredIPEnabled:(B
OOL)enable;

/// 直接从缓存中获取ipv6解析结果，无需等待.  如无缓存，或有缓存但已过期，并且enable为NO，则返回 nil
/// @param domain   域名
/// @param enable   是否允许返回过期ip
- (NSArray<NSString *> *)getIpv6ByCacheWithDomain:(NSString *)domain andExpiredIPEnabled:(B
OOL)enable;

///统计信息收集
-(NSArray *)getRequestReportInfo;
```

5. API使用示例

5.1 设置基本信息

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionar
y *)launchOptions {
   // Override point for customization after application launch.

   //唯一初始化方式
   DNSResolver *resolver = [DNSResolver share];
   //必传参数
   resolver.accountId = @"您在控制台注册应用时分配的Account ID";

   return YES;
}
```

5.2 域名解析接口

SDK提供了多种域名解析方法供用户选择和使用，可在 `DNSResolver.h` 头文件中查看。 下面以其中一个自动区分网络环境（ipv4-only、ipv6-only、ipv4和ipv6双栈）的解析方法为例，进行说明。

接口声明：

```
/// 获取域名解析后的ip数组，自动区分网络环境（ipv4-only、ipv6-only、ipv4和ipv6双栈）得到适用于当前网络环境的ip
/// @param domain            域名
/// @param complete        回调(所有ip地址)
- (void)getIpsDataWithDomain:(NSString *)domain complete:(void(^)(NSArray<NSString *> *dataArray))complete;
```

接口调用示例：

```
[[DNSResolver share] getIpsDataWithDomain:@"www.taobao.com" complete:^(NSArray<NSString *> *dataArray) {
    //dataArray为域名www.taobao.com所对应的IP地址数组
   if (dataArray.count > 0) {
       //TODO: 使用IP地址进行url连接
   }
}];
```

5.3 直接从缓存中获取解析结果

接口声明：

```
/// 自动区分网络环境（ipv4-only、ipv6-only、ipv4和ipv6双栈）直接从缓存中获取适用于当前网络环境的ip数组，无需等待.  如无缓存，或有缓存但已过期，返回 nil
/// @param domain    域名
/// @param enable    是否允许返回过期ip
- (NSArray<NSString *> *)getIpsByCacheWithDomain:(NSString *)domain andExpiredIPEnabled:(BOOL)enable;
```

调用示例：

```
NSArray *result = [[DNSResolver share] getIpsByCacheWithDomain:@"域名" andExpiredIPEnabled:YES];
//拿到缓存结果
if (result.count > 0) {
   //TODO: 使用ip地址进行url连接

}
```

注意：直接从缓存中拿缓存结果，速度较快，但无缓存、或者有缓存但缓存的解析结果已过期并且enable为NO时，返回结果会为nil。

5.4 统计信息收集

接口声明：

```
///统计信息收集
-(NSArray *)getRequestReportInfo;
```

调用示例：

```
NSArray *array = [[DNSResolver share] getRequestReportInfo];
```

数据格式：

```
(
    {
        avgRtt = "1";                                   // 域名平均解析时间ms
        cacheDnsNum = 0;                                // 命中缓存次数
        domain = "www.taobao.com";                      // 解析的域名
        gobackLocaldnsNum = 0;                          // 降级到LocalDNS的次数
        localErro = 0;                                  // LocalDNS解析失败次数
        maxRtt = "60";                                  // 域名最大解析时间ms
        noPermissionErro = 0;                           // 用户鉴权失败次数
        noResponseErro = 0;                             // 请求超时无应答次数
        requestPDnsNum = 1;                             // 递归查询的次数
        sp = 中国移动;                                  // 运营商名字
        successNum = 1;                                 // 解析成功次数
        timeoutErro = 0;                                // 网络超时错误次数
        type = 28;                                      // ip类型1代表ipv4、28代表ipv6
        urlParameterErro = 0;                           // 请求参数格式错误次数
        urlPathErro = 0;                                // URL错误次数
    }
        ......
);
```

6.注意事项

1. `pdns-sdk-ios.framework` 支持最低版本为iOS9.0。
2. 使用http协议请求时，需要在 `Info.plist` 中设置

   `App Transport Security Settings->Allow Arbitrary Loads` 为 `YES` 。
3. 通过阿里公共DNS获得域名的IP地址后，客户端可以使用这个IP发送业务请求，HTTP请求头的Host字段需改为原来的域名。

   例如：

   ```
   //ip为原域名解析后的ip地址
   NSURL *url = [NSURL URLWithString:[NSString stringWithFormat:@"https://%@", ip]];
   NSMutableURLRequest *mutableReq = [NSMutableURLRequest requestWithURL:url cachePolicy:N
   SURLRequestUseProtocolCachePolicy timeoutInterval: 10];
   //设置host
   [mutableReq setValue:@"原域名" forHTTPHeaderField:@"host"];
   ```
4. 为帮助用户更快的使用阿里公共DNS iOS SDK，我们为开发者提供Demo程序，您可以

   下载到本地作为参考。请点击这里，下载Demo程序。