ALIBABA CLOUD

阿里云

HTTPDNS

最佳实践

文档版本：20220104

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.Unity框架最佳实践 05
2.如何使用“会话追踪方案”排查解析异常 09
3.如何利用HTTPDNS降低DNS解析开销 12
4.HTTPDNS域名解析场景下如何使用Cookie? 15
5.C/C++语言业务场景实现IP直连 16

1.Unity框架最佳实践

Unity是实时3D互动内容创作和运营平台。包括游戏开发、美术、建筑、汽车设计、影视在内的所有创作者，借助Unity将创意变成现实。Unity平台提供一整套完善的软件解决方案，可用于创作、运营和变现任何实时互动的2D和3D内容，支持平台包括手机、平板电脑、PC、游戏主机、增强现实和虚拟现实设备。

随着使用Unity开发的移动应用越来越多，应用类型越来越广泛，我们专门提供了针对Unity的HTTPDNS插件，方便Unity开发者以熟悉方式接入HTTPDNS功能。

环境说明

Unity 2019.4.21f1c1

项目说明

项目地址：下载地址

本项目是一个unity demo项目，Assets/Plugins目录下为插件代码。

集成操作

1. 复制Assets/Plugins到unity项目

目前插件直接以文件的方式提供，后续再考虑封装为Unity Package。

> 注意
>
> 插件使用了Custom Main Gradle TemplateCustom Base Gradle TemplateCustom Proguard File，如果项目有其他地方也需要定制这些文件，请注意合并修改内容。

配置

1. 在使用httpdns之前请先初始化。

```
// 初始化需要在平台申请的accountId，如果开启了鉴权，需要填写secret，如果没有开启可以传空字符串
HttpDnsHelper.init("accountId", "secret");
```

2. 其他一些常用的配置：

```
        //开启debug日志
        HttpDnsHelper.debug(true);
        // 输出日志
        HttpDnsHelper.setLogger((log) => Debug.Log(log));
        // 开启网络变化自解析
        HttpDnsHelper.setPreResolveAfterNetworkChanged(true);
        // 允许使用过期IP
        HttpDnsHelper.setExpiredIPEnabled(true);
        // 本地缓存IP
        HttpDnsHelper.setCachedIPEnabled(true);
        List<string> list = new List<string>();
        list.Add("www.taobao.com");
        // 预解析域名
        HttpDnsHelper.setPreResolveHosts(list);
        // 使用https请求解析域名
        HttpDnsHelper.setHttpsRequestEnabled(true);
        // 配置域名解析请求的超时时长
        HttpDnsHelper.setTimeoutInterval(3000);
```

API

接口类为Assets/Plugins/HttpDnsHelper.cs。

调用示例请参考Assets/DownloadListener.cs。

主要方法如下：

init初始化

必须先初始化，才能正常使用。

```
public static void init(string accountId, string secret)
```

参考原生API说明AndroidiOS。

debug

启用debug日志。

```
public static void debug(bool enable)
```

setLogger

设置日志接收接口。

```
public static void setLogger(Action<string> logger)
```

setPreResolveAfterNetworkChanged

网络变化后，自动解析域名。

```
 public static void setPreResolveAfterNetworkChanged(bool enable)
```

参考原生API说明AndroidiOS。

setExpiredIPEnabled

允许使用ttl过期IP。

```
public static void setExpiredIPEnabled(bool enable)
```

参考原生API说明AndroidiOS。

setCachedIPEnabled

开启本地缓存。

```
public static void setCachedIPEnabled(bool enable)
```

参考原生API说明AndroidiOS。

setPreResolveHosts

设置预解析的域名，type取值说明如下：

1：表示解析ipv4地址；2：表示解析ipv6地址；3：表示ipv4、ipv6都需要解析。

```
public static void setPreResolveHosts(List<string> hosts, int type)
```

参考原生API说明AndroidiOS。

setAuthCurrentTime

校正本地时间，避免鉴权失败。

```
public static void setAuthCurrentTime(long time)
```

参考原生API说明AndroidiOS。

setHttpsRequestEnabled

开启使用HTTPS接口解析域名，默认使用http。

```
public static void setHttpsRequestEnabled(bool enable)
```

参考原生API说明Android。

setTimeoutInterval

设置域名解析请求的超时时间。

```
public static void setTimeoutInterval(int time)
```

参考原生API说明AndroidiOS。

setRegion

设置region节点，调用后，会按照region更新服务IP。

```
public static void setRegion(string region)
```

参考原生API说明AndroidiOS。

getIpsByHostAsync

解析域名ipv4地址。

```
public static List<string> getIpsByHostAsync(string host)
```

参考原生API说明Android iOS。

getIpv6sByHostAsync

解析域名ipv6地址。

```
public static List<string> getIpv6sByHostAsync(string host)
```

参考原生API说明Android。

setIPProbeList

设置IP优选域名。

```
public static void setIPProbeList(List<ProbeItem> probeItems)

public class ProbeItem
{
        public string host;
        public int port;
}
```

参考原生API说明Android iOS。

setDegradationFilter

设置域名的解析过滤逻辑。

```
public static void setDegradationFilter(Func<string, bool> filter)
```

参考原生API说明Android iOS。

getSessionId

获取用户sessionId。

```
public static string getSessionId()
```

参考原生API说明Android iOS。

enableIpv6

开启ipv6解析。

```
public static void enableIpv6(bool enable)
```

参考原生API说明iOS。

2.如何使用“会话追踪方案”排查解析异常

当您对解析结果或HTTPDNS服务质量产生如下疑问，希望排查问题时，请参考此“会话追踪方案”。

- 您认为解析出的IP不符合预期，以至于影响了业务域名的服务质量。
- 您接入的APM系统显示，HTTPDNS服务质量有问题。

> 注意
>
> - 对于有疑问的解析IP，需要先确定是否由HTTPDNS解析的结果：需要提供从HTTPDNS接口取出IP的日志及具体的HTTPDNS接入代码。
> - 由于技术限制，我们只提供三天内的解析问题排查。
> - 由于此方案需要配合SDK的 `getSessionId` 方法，您至少需要升级至 `Android ≥ 1.2.3` ，`iOS ≥ 1.6.20` 才能应用此方案。

背景介绍

对于一个配置了分线路解析的域名，根据客户入网IP的不同，返回的IP是根据各线路的配置而不同。为了获得更好的服务质量，服务节点调度系统（GSLB）可能会频繁地变更上述的线路调度配置，例如CDN系统，可能每几个小时就会调整一次调度配置。如果客户端获取到了错误的调度结果（即解析结果），可能会导致服务质量受损，也就是慢或超时，还有可能导致服务不可用。

为了定位上述问题，一般需要您提供来源IP、解析时间、解析的域名、解析结果IP列表来定位异常现场，以便观察是否有对应的调度结果记录。但这种定位异常现场的方案存在以下问题：

- 客户端难以提供准确的入网IP。
- 即使提供了准确的入网IP，由于IP一般是共享的，不一定能定位到该次解析现场。
- 对于同一个网络环境，运营商可能会针对不同的目的IP，提供不同的入网IP。
- 在App的一次生命周期中，入网IP可能发生变化。
- 难以解释以上入网IP发生的变化原因。

引入sessionId（会话唯一ID）

sessionId参数在App启动时生成，在整个App生命周期中不会发生变化。在同一个App生命周期中的HTTPDNS解析请求均会携带同一个sessionId，服务器则会记录这个参数，并生成索引。与App异常日志上报中以单个设备ID为索引条件类似，本方案以App的一次生命周期作为做索引条件。与问题常规排查方案中“来源IP-解析结果”组合的追踪方式相比，使用sessionId对App生命周期进行追踪，是一种更精准，更适用于App场景的问题排查方式。

通过引入sessionId，我们不再需要您提供客户端入网IP，改为根据sessionId获取准确的IP信息。同时可以追踪在App的一次生命周期中所有的HTTPDNS解析请求，可以观测到入网IP可能发生的变化，还可以根据net参数的变化解读客户入网IP变化的原因。

例如：一个客户通过4G入网，接入使用的SIM卡可能是广东的运营商发行的，那么接入IP可能是广东区域的。当从HTTPDNS获取IP后，客户连接了其物理所在地的有线网络或wifi，可能导致接入IP立即变为该地区的IP，导致内容服务质量受到影响。这种场景在传统LocalDNS环境下和HTTPDNS下都会出现。

使用“会话追踪方案”

由于App更新升级过程是渐进式的，无法在短时间内部署最新的版本，我们建议您在接入HTTPDNS时就参考本方案，记录解析结果和对应的sessionId，以便后续有问题时，方便准确提供追查线索。

- SDK接入：我们会在App启动时生成sessionId，并提供了getSessionId方法获取生成的结果。您需要更新SDK版本来使用此功能（Android ≥ 1.2.3, iOS ≥ 1.6.20），但我们建议您升级至最新版本，以便解决一些已知的其他SDK问题。
- HTTP API方式接入（即非SDK接入）：您可以仿照SDK的逻辑生成相应参数。
- 端日志上报调度服务质量问题时，需要额外记录sessionId参数。

另外，您也可以将sessionId携带至其他系统中。例如当使用HTTPDNS解析结果，请求该域名的服务时，可以在URL中额外携带sessionId参数，这样可以更加完整地观察客户使用场景。

在本方案中，每次SDK对HTTPDNS服务器的请求中，我们添加了如下三个参数（服务器会记录这些信息）：

- sid=<sessionId:[a-zA-Z0-9]{12}>，在SDK启动时生成，用于标记一次独立的App生命周期。
- net=<4g|3g|2g|wifi|unknown>，用于标记请求发起时刻os层提供的网络情况；提供sessionId后，我们可以将当时记录的net信息反馈给您。
- bssid=<wifi_bssid>，用于标记不同的wifi网络；提供sessionId后，我们可以将当时记录的bssid信息反馈给您。
- 示例URL:
  http://203.107.1.33/100000/dhost=www.aliyun.com&sid=wInhNA3iM0PK&net=wifi&bssid=54e061553e79

排查场景：解析出的结果不符合预期

当出现解析出的结果不符合预期，您认为可能和HTTPDNS有关，需要排查时：

请您提交HTTPDNS工单，并提供以下信息：

- 解析时的sessionId、解析时间、解析的域名、解析结果、预期解析结果。
- 是否造成了实际业务服务质量下降（例如调度到了不正确地域的服务节点）？
- 是否造成了业务不可用的问题？

如果发现这类问题，建议您尽早针对此类情况，考虑重新获取域名解析结果（HTTPDNS或LocalDNS），并进行业务重试。

我们会根据sessionId提供以下调查结果：

- 该解析结果是否由HTTPDNS服务提供？

根据端上策略不同，解析结果可能有三种不同的来源：HTTPDNS、LocalDNS、持久化缓存。

- 该次HTTPDNS解析对应的网络环境及入网IP。
- 客户在该次App生命周期内是否存在IP跨区域切换的情况。

排查场景：APM显示HTTPDNS服务质量有问题

当您接入了APM组件（即线上版端性能分析工具），APM组件中显示HTTPDNS服务出现无法连接的问题时

请提交HTTPDNS工单，并提供以下信息：

APM系统截图：连接失败或返回错误时的URL地址、出现的错误信息、以及问题出现的时间

- URL中会包含sessionId、网络情况等信息。
- 虽然该请求可能没有到达服务器，但请求时的网络情况（net）会被添加到URL中，并由您接入的APM系统记录。

我们会根据您提供的信息提供如下帮助：

- 协助您分析该次失败请求的原因。
- 该次App生命周期内的相关请求记录，以便做如下分析：
  - 判断是否存在网络切换的情况。

- 是否同一时间有成功的请求。

3.如何利用HTTPDNS降低DNS解析开销

背景说明

移动场景下DNS的解析开销是整个网络请求延迟中不可忽视的一部分。一方面基于UDP的localDNS解析在高丢包率的移动网络环境下更容易出现解析超时的问题，另一方面在弱网环境下DNS解析所引入的动辄数百毫秒的网络延迟也大幅加重了整个业务请求的负担，直接影响用户的终极体验。

解决方案

HTTPDNS在解决了传统域名劫持以及调度精确性的问题的同时，也提供了开发者更灵活的DNS管理方式。通过在客户端合理地应用HTTPDNS管理策略，我们甚至能够做到DNS解析0延迟，大幅提升弱网环境下的网络通讯效率。

DNS解析0延迟的主要思路包括：

- 构建客户端DNS缓存

  通过合理的DNS缓存，我们确保每次网络交互的DNS解析都是从内存中获取IP信息，从而大幅降低DNS解析开销。根据业务的不同，我们可以制订更丰富的缓存策略，如根据运营商缓存，可以在网络切换的场景下复用已缓存的不同运营商线路的域名IP信息，避免网络切换后进行链路重选择引入的DNS网络解析开销。另外，我们还可以引入IP本地化离线存储，在客户端重启时快速从本地读取域名IP信息，大幅提升首页载入效率。

- 热点域名预解析

  在客户端启动过程中，我们可以通过热点域名的预解析完成热点域名的缓存载入。当真正的业务请求发生时，直接由内存中读取目标域名的IP信息，避免传统DNS的网络开销。

- 懒更新策略

  绝大多数场景下业务域名的IP信息变更并不频繁，特别是在单次APP的使用周期内，域名解析获取的IP往往是相同的（特殊业务场景除外）。因此我们可以利用DNS懒更新策略来实现TTL过期后的DNS快速解析。所谓DNS懒更新策略即客户端不主动探测域名对应IP的TTL时间，当业务请求需要访问某个业务域名时，查询内存缓存并返回该业务域名对应的IP解析结果。如果IP解析结果的TTL已过期，则在后台进行异步DNS网络解析与缓存结果更新。通过上述策略，用户的所有DNS解析都在与内存交互，避免了网络交互引入的延迟。

Demo示例

我们在HTTPDNS Demo github中提供了Android SDK以及HTTPDNS API接口的使用例程，这里我们通过使用Android SDK的例程演示如何实现0延迟的HTTPDNS服务。

```
public class NetworkRequestUsingHttpDNS {

    private static HttpDnsService httpdns;
    // 填入您的HTTPDNS accoutID信息，您可以从HTTPDNS控制台获取该信息
    private static String accountID = "100000";
    // 您的热点域名
    private static final String[] TEST_URL = {"http://www.aliyun.com", "http://www.taobao.com"}
;

    public static void main(final Context ctx) {
        try {
            // 设置APP Context和Account ID，并初始化HTTPDNS
            httpdns = HttpDns.getService(ctx, accountID);
            // DegradationFilter用于自定义降级逻辑
            // 通过实现shouldDegradeHttpDNS方法，可以根据需要，选择是否降级
            DegradationFilter filter = new DegradationFilter() {
                @Override
                public boolean shouldDegradeHttpDNS(String hostName) {
```

```
                // 此处可以自定义降级逻辑，例如www.taobao.com不使用HttpDNS解析
                // 参照HttpDNS API文档，当存在中间HTTP代理时，应选择降级，使用Local DNS
                return hostName.equals("www.taobao.com") || detectIfProxyExist(ctx);
            }
        };
        // 将filter传进httpdns，解析时会回调shouldDegradeHttpDNS方法，判断是否降级
        httpdns.setDegradationFilter(filter);
        // 设置预解析域名列表，真正使用时，建议您将预解析操作放在APP启动函数中执行。预解析操作为异步行为，不会阻塞您的启动流程
        httpdns.setPreResolveHosts(new ArrayList<>(Arrays.asList("www.aliyun.com", "www.taobao.com")));
        // 允许返回过期的IP，通过设置允许返回过期的IP，配合异步查询接口，我们可以实现DNS懒更新策略
        httpdns.setExpiredIPEnabled(true);

        // 发送网络请求
        String originalUrl = "http://www.aliyun.com";
        URL url = new URL(originalUrl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        // 异步接口获取IP，当IP TTL过期时，由于采用DNS懒更新策略，我们可以直接从内存获得最近的DNS解析结果，同时HTTPDNS SDK在后台自动更新对应域名的解析结果
        ip = httpdns.getIpByHostAsync(url.getHost());
        if (ip != null) {
            // 通过HTTPDNS获取IP成功，进行URL替换和HOST头设置
            Log.d("HTTPDNS Demo", "Get IP: " + ip + " for host: " + url.getHost() + " from HTTPDNS successfully!");
            String newUrl = originalUrl.replaceFirst(url.getHost(), ip);
            conn = (HttpURLConnection) new URL(newUrl).openConnection();
        }
        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.e("HTTPDNS Demo", "Response: " + response.toString());

    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 * 检测系统是否已经设置代理，请参考HttpDNS API文档。
 */
public static boolean detectIfProxyExist(Context ctx) {
    boolean IS_ICS_OR_LATER = Build.VERSION.SDK_INT >= Build.VERSION_CODES.ICE_CREAM_SANDWICH;
    String proxyHost;
    int proxyPort;
    if (IS_ICS_OR_LATER) {
        proxyHost = System.getProperty("http.proxyHost");
        String port = System.getProperty("http.proxyPort");
        proxyPort = Integer.parseInt(port != null ? port : "-1");
    } else {
        proxyHost = android.net.Proxy.getHost(ctx);
```

```
            proxyPort = android.net.Proxy.getPort(ctx);
        }
        return proxyHost != null && proxyPort != -1;
    }
}
```

对于使用HTTPDNS API接口的开发者，您可以在客户端自己定制更高效，并且符合您需求的HTTPDNS管理逻辑。

4.HTTPDNS域名解析场景下如何使用Cookie？

请参考云栖社区文档HTTPDNS域名解析场景下如何使用Cookie？

5.C/C++语言业务场景实现IP直连

移动研发平台 EMAS的HTTPDNS服务，目前仅支持Android/iOS端的SDK接入。其他类型终端，可通过C/C++语言curl库实现IP直连的方式，使用HTTPDNS服务。

背景知识

使用IP直连访问HTTPDNS时，主要处理以下关键点：

- HTTP Host头设置。
- HTTPS SNI设置。
- HTTPS 证书校验处理。

前提条件

使用curl库。

操作步骤

1、通过HTTP API方式解析IP，具体操作请参见：单域名解析接口。

2、使用C/C++ curl库的Custom addresses for hosts功能，进行IP直连请求，具体内容请参见：libcurl库文档。

应用示例

假设业务环境如下：

- 访问业务网站：https://example.com/
- 域名：example.com
- IP地址：93.184.216.34

IP直连请求如下：

- 基于curl命令行

```
$ curl --max-time 5 --resolve example.com:443:93.184.216.34 https://example.com/
```

- 基于libcurl库

```
#include <stdio.h>
#include <curl/curl.h>

int main(void) {
    CURL *curl;
    CURLcode res;

    curl_global_init(CURL_GLOBAL_DEFAULT);

    curl = curl_easy_init();
    if (curl) {
        curl_easy_setopt(curl, CURLOPT_URL, "https://example.com/");
        curl_easy_setopt(curl, CURLOPT_TIMEOUT, 5);

        /* 设置域名和通过HTTPDNS预先解析出来的IP */
        struct curl_slist *dns;
        dns = curl_slist_append(NULL, "example.com:443:93.184.216.34");
        curl_easy_setopt(curl, CURLOPT_RESOLVE, dns);

        /* Perform the request, res will get the return code */
        res = curl_easy_perform(curl);
        if (res != CURLE_OK) {
            fprintf(stderr, "curl_easy_perform() failed: %s\n",
                    curl_easy_strerror(res));
        }

        /* always cleanup */
        curl_easy_cleanup(curl);
    }

    curl_global_cleanup();

    return 0;
}
```