

# 目录

前言	1.1
Protobuf概览	1.2
Protobuf正向	1.3
Protobuf逆向	1.4
普通的iOS类	1.4.1
GPBMessage	1.4.1.1
GPBDescriptor	1.4.1.2
GPBFieldDescriptor	1.4.1.3
lite的C++类	1.4.2
MessageLite	1.4.2.1
YouTube中Protobuf	1.4.3
普通的iOS类	1.4.3.1
YTPlayerRequest	1.4.3.1.1
YTAdBreakRequest	1.4.3.1.2
YTInnertubeContext	1.4.3.1.3
YTClientInfo	1.4.3.1.4
lite的C++类	1.4.3.2
OnesieRequestProto	1.4.3.2.1
附录	1.5
参考资料	1.5.1

# iOS逆向YouTube: protobuf逆向

- 最新版本: v0.7
- 更新时间: 20221104

## 简介

整理iOS逆向YouTube期间涉及到的Protobuf的逆向。先是概览；再是Protobuf的正向相关知识；然后是Protobuf逆向内容，包括普通的iOS的类、lite精简版的C++的类、YouTube中Protobuf的内容；其中，普通的iOS的类有GPBMessage、GPBDescriptor、GPBFieldDescriptor；lite精简版C++的类有MessageLite；YouTube中的普通的iOS类的Protobuf类有：YTIPlayerRequest、YTAdBreakRequest、YTInnerTubeContext、YTIClientInfo，以及lite的C++类有OnesieRequestProto；

## 源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

### HonKit源码

- [crifan/ios\\_re\\_protobuf\\_reverse: iOS逆向YouTube: protobuf逆向](#)

如何使用此HonKit源码去生成发布为电子书

详见：[crifan/honkit\\_template: demo how to use crifan honkit template and demo](#)

### 在线浏览

- [iOS逆向YouTube: protobuf逆向 book.crifan.org](#)
- [iOS逆向YouTube: protobuf逆向 crifan.github.io](#)

### 离线下载阅读

- [iOS逆向YouTube: protobuf逆向 PDF](#)
- [iOS逆向YouTube: protobuf逆向 ePub](#)
- [iOS逆向YouTube: protobuf逆向 Mobi](#)

## 版权和用途说明

此电子书教程的全部内容，如无特别说明，均为本人原创。其中部分内容参考自网络，均已备注了出处。如发现有侵权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

各种技术类教程，仅作为学习和研究使用。请勿用于任何非法用途。如有非法用途，均与本人无关。

## 鸣谢

感谢我的老婆陈雪的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

## 更多其他电子书

本人 `crifan` 还写了其他 150+ 本电子书教程，感兴趣可移步至：

[crifan/crifan\\_ebook\\_readme: Crifan的电子书的使用说明](#)

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：

2022-11-04 15:59:00

# Protobuf概览

在iOS逆向YouTube期间，遇到了 Protobuf 的逆向。

其中涉及到，了解Protobuf的正向内容，和相关逆向内容。

crifan.org，使用署名4.0国际(CC BY 4.0)协议发布 all right reserved，powered by Gitbook最后更新：

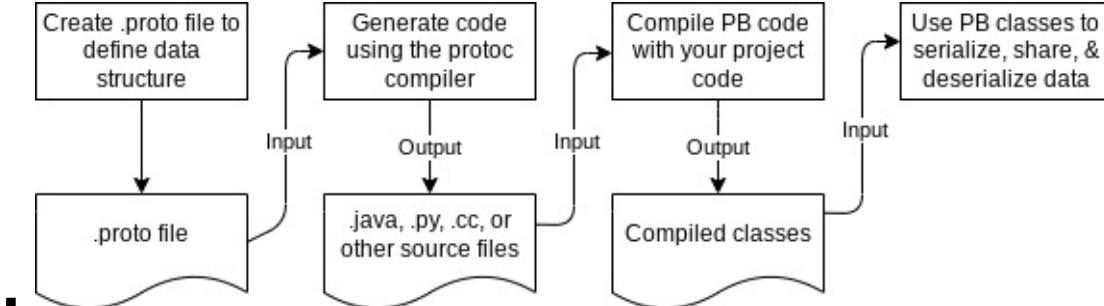
2022-11-04 14:45:51

# Protobuf正向

TODO:

- 【整理】 Chromium相关: Protobuf源码和文档
  - 【整理】 Protocol Buffers中.proto文件写法即protocobuf的语法
  - 【已解决】 protobuf中字段规则类型: required和optional
- 

- `protobuf`
    - `protobuf = Protocol Buffers`
    - 概述
      - Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data
    - 特点
      - 比JSON、XML等格式，占用体积更小，更快，更简单
    - 是什么：是一系列的组合 = 包含三部分
      - 定义语言 : `definition language`
      - 自己写的: `.proto`文件
      - 代码 = `code`
        - 需要专门的protocol compiler (对应命令行工具是: `protoc`)去编译生成对应语言的代码
          - 用于操作数据的代码
        - 运行时 : `language-specific protobuf runtime libraries`
        - 序列化 : `serialization format for data`
          - 写入文件
          - 或: 用网络传输
          - 注:
            - 相关名词
              - `序列化 = serialize = encode = 编码`
              - `反序列化 = deserialize = parse = 解析 = decode = 解码`
  - 支持多种语言
    - `proto2`
      - Java
      - Python
      - Objective-C
      - C++
      - PHP
    - `proto3`
      - Kotlin
      - Dart
      - Go
      - Ruby
      - C#
- 支持场景

- 临时短期的网络传输
- 长期的数据保存
- 特点
  - 支持扩展extended=extensions
    - Protocol buffers can be extended with new information without invalidating existing data or requiring code to be updated
    - In proto2, messages can allow extensions to define fields outside of the message, itself. For example, the protobuf library's internal message schema allows extensions for custom, usage-specific options.
  - 兼容性好
    - 向后兼容性很好
    - 向前兼容性也很好
      - 无缝的支持:
        - 字段的变化
        - 新增字段
        - 删除字段
- 使用Protocol Buffers的好处
  - 体积小=占用空间少=数据存储很紧凑
  - 解析速度快
  - 支持多种语言
  - 有各种经过优化的功能（通过编译器生成的类去实现的）
- protobuf工作流 Protocol buffers workflow
 

```

graph LR
    A[Create .proto file to define data structure] --> B[.proto file]
    B -- Input --> C[Generate code using the protoc compiler]
    C -- Output --> D[java, .py, .cc, or other source files]
    D -- Input --> E[Compile PB code with your project code]
    E -- Output --> F[Compiled classes]
    F -- Input --> G[Use PB classes to serialize, share, & deserialize data]
  
```

该图展示了protobuf的工作流程。首先，创建一个定义数据结构的.proto文件。然后，使用protoc编译器生成相应的语言代码（如.java, .py, .cc等）。接着，将这些代码与项目代码一起编译成字节码（Compiled classes）。最后，使用这些字节码来序列化、共享和反序列化数据。

- proto compiler = protoc
  - 输入: .proto
  - 输出: 对应的特定语言的代码
    - 对每个字段和方法, 都有一个accessor访问器
    - 用于在 数据结构 和 二进制数据 之间解析和转换
  - 编译生成的文件
    - C++: .h和.cc
    - Java: .java
    - Kotlin: .kt
    - Python: 生成内容是一个模块module, 放在.proto文件中
    - Go: .pb.go
    - Ruby: .rb
    - Objective-C: pbobjc.h和pbobjc.m
    - C#: .cs
    - Dart: .pb.dart
- 相关

- google内部常用到的一些protobuf定义
  - timestamp
    - protobuf/timestamp.proto at main · protocolbuffers/protobuf (github.com)
  - status
    - googleapis/status.proto at master · googleapis/googleapis (github.com)
- 文档和代码
  - 详见：
    - 【整理】 Chromium相关： Protobuf源码和文档

## protobuf举例

.proto 定义：

```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;
}
```

-》 序列化： Java 写入文件：

```
Person john = Person.newBuilder()
  .setId(1234)
  .setName("John Doe")
  .setEmail("jdoe@example.com")
  .build();
output = new FileOutputStream(args[0]);
john.writeTo(output);
```

-》 反序列化： C++解析

```
Person john;
fstream input(argv[1],
  ios::in | ios::binary);
john.ParseFromIstream( input );
id = john.id();
name = john.name();
email = john.email();
```

更多例子：

- examples - external/github.com/google/protobuf - Git at Google (googlesource.com)

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：  
2022-11-04 15:38:42

# Protobuf逆向

TODO:

- 【记录】Protobuf的full和lite的Message正向和逆向机制对比

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:  
2022-11-04 14:50:13

## 普通的iOS类

TODO:

- 【未解决】Protobuf的iOS的ObjC版的lite的库即runtime运行时
- 【未解决】iOS的ObjC的protobuf的sample示例的正向编译和逆向尝试
- 【已解决】protobuf逆向：已有data如何获取子属性子字段的数据
- 【已解决】protobuf逆向：iOS端无需data直接解析protobuf类的字段定义

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：  
2022-11-04 15:43:08

# GPBMessage

TODO:

- 【未解决】研究YouTube逻辑: GPBMessage的数据
  - 【已解决】研究YouTube逻辑: GPBMessage
  - 【已解决】研究YouTube逻辑: 关于GPBMessage子类中protobuf属性字段和顺序的相关理解
  - 【未解决】研究YouTube逻辑: GPBMessage的parseFromData
- 

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:  
2022-11-04 15:32:00

# GPBDescriptor

TODO:

- 【未解决】研究YouTube逻辑: GPBDescriptor

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:  
2022-11-04 14:49:13

## GPBFieldDescriptor

TODO:

- 【已解决】研究YouTube逻辑: GPBFieldDescriptor

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:  
2022-11-04 14:52:38

## lite的C++类

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：  
2022-11-04 15:43:02

# MessageLite

TODO:

- 【整理】Protobuf中的MessageLite

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:  
2022-11-04 14:53:39

# YouTube中Protobuf

TODO:

- 【已解决】研究YouTube逻辑: protobuf中搜索内容对应的可能的字符串编码
- 【未解决】研究YouTube逻辑: protobuf的逆向

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:  
2022-11-04 14:53:25

## 普通的iOS类

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：  
2022-11-04 15:44:28

## YTIPPlayerRequest

TODO:

- 【未解决】研究YouTube逻辑: YTIPPlayerRequest的descriptor
- 【未解决】研究YouTube逻辑: YTIPPlayerRequest的context的protobuf的number不一致
- 【已解决】研究YouTube逻辑: 通过GPBFieldDescriptor调试出YTIPPlayerRequest的protobuf的属性字段定义

推导出 YTIPPlayerRequest 的 protobuf 的字段定义, 大概类似于:

```
message YTIPPlayerRequest {
    YTITubeContext context = 1;
    NSString *videoId = 2;
    _Bool contentCheckOk = 3;
    YTIPPlaybackContext playbackContext = 4;
    _Bool racyCheckOk = 5;
    NSString *id_p = 6;
    NSString *t = 7;
    _Bool forOffline = 8;
    NSString *playlistId = 9;
    int playlistIndex = 10;
    unsigned int startTimeSecs = 11;
    NSString *params = 12;
    ??? = 13;
    NSData *offlineSharingWrappedKey = 14;
    GPBInt32Array *installedSharingServiceIdsArray = 15;
    YTIPPlayerAttestationRequestData *attestationRequest = 16;
    NSString *referringApp = 17;
    NSString *referrer = 18;
    NSString *serializedThirdPartyEmbedConfig = 19;
    _Bool proxiedByOnesie = 20;
    ??? = 21;
    NSString *hostAppToken = 22;
    NSString *cpn = 23;
    ??? = 24;
    _Bool overrideMutedAtStart = 25;
    YTIPPlayerRequestCaptionParams *captionParams = 26;
    ??? = 27;
    YTIPPlayerRequestVideoQualitySettingParams *videoQualitySettingParams = 28;
}
```

## YTIAdBreakRequest

TODO:

- 【已解决】研究YouTube逻辑：从data解析出YTIAdBreakRequest所有的字段属性的值
- 【未解决】研究YouTube逻辑：获取YTIAdBreakRequest所有的字段的定义即name和number映射关系
- 【已解决】研究YouTube逻辑：protobuf类YTIAdBreakRequest

想要从protobuf的类YTIAdBreakRequest序列化后的NSData数据data，去解析出：

对应的所有属性的值，包括嵌套的子属性的值

思路是：

YTIAdBreakRequest 的基类=父类是 GPBMessage，其中有Class级别的函数：

```
+ (id)parseFromData:(id)arg1;
```

所以可以直接拿来解析。

具体写法：

```
po [objc_getClass("YTIAdBreakRequest") parseFromData: newHttpBodyData]
```

即可得到：

整个protobuf类YTIAdBreakRequest的所有属性的值，包括嵌套的属性。

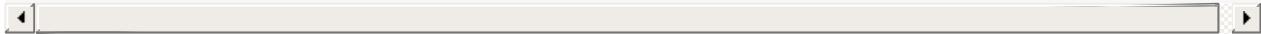
举例：

- 输入：二进制的数据

```
(lldb) po newHttpBodyData
0ab3160a 9b120a05 7a682d43 4e120243 4e520243 4e620541 70706c65 6a096950 686f6e65 392c3180 0105
8a01 0731372e 30382e32 92010369 4f539a01 0c31332e 332e312e 31374435 30a802f7 02b0029b 05c80202
f00201b8 03f702c0 039b05e8 0303f203 830e0ab8 04434d61 5a714a6f 47454f72 4b726755 51714d2d 6f467
844 55673634 46455065 49726755 5134726d 75425244 53334b30 46454f71 64725155 516d6361 75425244 4
66d7134 46454d61 46725155 516c3943 75425243 44304b34 4645496a 70725155 51754975 75425244 746971
30 46454b58 76725155 51326275 74425243 476f6134 46454f62 4e726755 516c4d2d 75425244 726d6130 46
45497a 79725155 51703532 75425244 586b6130 46454a6a 55725155 51786232 75425243 43746134 46454f5
4 4b726755 51676357 74425244 667a7134 46454d65 78725155 51324c79 75425244 58396130 46454e76 4b7
26755 51373869 75425244 716c3677 4645496d 78726755 51676269 75425244 377a7134 46454a61 61725155
516d7236 75425244 316c7134 46454947 47726755 51305f47 74425243 626f4b34 46454b48 39725155 516f
4c6d 75425244 4e344b30 46454a50 51726755 516e7632 74425244 72354b30 46454961 31726755 51367271
74425243 47793634 46454f54 4e726755 51394d65 74425243 6c734b34 46476a4a 42533342 6c5a4768 35545
456 30566b4e 31515739 32633031 54556d52 75616a64 69624638 30526d68 454e6d70 4d576e5a 55593245 3
5525852 534f4768 47566b52 52515349 79515574 775a5752 6f655530 3164465a 44645546 76646e4e 4e5531
4a 6b626d6f 33596d78 664e455a 6f52445a 71544670 3256474e 684f5556 30556a68 6f526c5a 45555545 71
47454e 4254564e 45515442 454e6b30 74634546 6f565551 3464444e 59524545 39505125 33442533 441adc0
```

4 434a5077 73706f47 45684d79 4f544134 4d544977 4e44677a 4e546b78 4e546b33 4e545530 474d615a 714  
 a6f47 4b4a6e47 7267556f 7a654374 42536947 74613446 4b4f544b 7267556f 6a504b74 42536a30 78363046  
 4b4b5777 7267556f 35733275 42536a5a 75363046 4b4a7567 7267556f 6f4c6d75 42536a72 6d613046 4b4a  
 5450 7267556f 67394375 42536961 76713446 4b4c694c 7267556f 6c394375 42536949 36613046 4b4e6938  
 7267556f 36703274 42536a31 6c713446 4b4e5344 7267556f 36706573 42536942 754b3446 4b4f724b 72675  
 56f 39346975 4253696f 7a366758 4b496d78 7267556f 78623275 42536a6b 7a613446 4b4e6552 7251556f 6  
 8737575 4253696e 6e613446 4b4a5051 7267556f 36727174 42536943 74613446 4b494846 7251556f 6f6632  
 74 42536947 6f613446 4b4e5078 7251556f 37597174 42536965 5f613046 4b4a6a55 7251556f 6c707174 42  
 536a47 68613046 4b494747 7267556f 70652d74 42536a53 334b3046 4b4e6631 7251556f 78374774 42536a4  
 6 6d713446 4b50764f 7267556f 37386975 42536a62 79713446 4b4e5f4f 7267556f 34726d75 42536a72 354  
 b3046 4d6a4a42 5333426c 5a476835 54545630 566b4e31 51573932 63303154 556d5275 616a6469 62463830  
 526d6845 4e6d704d 576e5a55 59324535 52585253 4f476847 566b5252 51546f79 51557477 5a57526f 6555  
 3031 64465a44 64554676 646e4e4e 55314a6b 626d6f33 596d7866 4e455a6f 52445a71 54467032 56474e68  
 4f555630 556a686f 526c5a45 55554643 47454e42 54564e45 51544245 4e6b3074 6345466f 56555134 64444  
 e59 52454539 50512533 44253344 2ae60443 4a507773 706f4745 6851784e 4449774f 544d354e 7a41784e 6  
 a417a4f 4467334e 5445784d 52695438 4c4b6142 69697969 5034534b 4c7a4b5f 52496f33 347a2d45 696941  
 67 7634534b 4f32455f 68496f38 395f3945 696a4c72 6630534b 4c364a5f 68496f31 594c2d45 69695369 76  
 34534b 4972675f 52496f33 6f332d45 696a4a68 5034534b 4b54455f 52496f6e 34662d45 69694138 5f30534  
 b 4d482d5f 52496f70 64443945 69695535 5077534b 4f4b5f5f 52496f67 594c3945 6969506a 5034534b 4e5  
 0655f 52496f6c 50443945 696a4b68 5f34534b 4d574f5f 68496f35 59442d45 696a4a2d 6630534b 4e47665f  
 52496f35 344c2d45 696a3533 7630534b 4f69435f 68496f33 4f443945 696a6872 5030534b 4e4c4c5f 5249  
 6f6e 765f3845 696a626b 5f30534b 4e794b5f 68496f77 49502d45 69693933 7630534b 5032735f 52496f79  
 2d7a3945 696a6267 5034534b 4b66385f 52496f71 72543945 696a4773 7630534b 4b6d715f 52496f6d 63623  
 945 69693876 7630534b 4a764d5f 52496f2d 49762d45 696a4f77 5030534b 4965435f 68496f6a 34582d45 6  
 9693539 6677534b 4e664d5f 52496f68 617a3945 696a4b32 5030534d 6a4a4253 33426c5a 47683554 545630  
 56 6b4e3151 57393263 30315455 6d527561 6a646962 46383052 6d68454e 6d704d57 6e5a5559 32453552 58  
 52534f 47684756 6b525251 546f7951 5574775a 57526f65 55303164 465a4464 55467664 6e4e4e55 314a6b6  
 2 6d6f3359 6d78664e 455a6f52 445a7154 46703256 474e684f 55563055 6a686f52 6c5a4555 55464348 454  
 e4254 564e4655 54424362 33526d4e 6b5a6164 33424755 55526b65 6a684a54 57356a64 3052533 448d0400  
 00004098 04e003f0 04018205 0d417369 612f5368 616e6768 6169a205 800320e5 89ffcfd9 ab8cfb92 0120  
 83f8 d594f8d8 aed25620 d5c1ccf7 9387d7fb 5520be84 878c8afe f0c4e701 20d9dd90 cccc848f f3800120  
 dade94fc aaacb990 5920c6ba 9597b4f9 c9b36120 86f2abaa 94cf4eed 2020c5c5 9488cddf ff879a01 20e2a  
 1c0 b0e6df97 f22f209d a1cda588 d4fcf07e 20fd8ec3 a2f8c5c3 d00320d9 a899e6ad 94dbf91f 2095a5f6 c  
 d83f3ff 9a1420dc ed99fc2 cfde93da 0120b498 bdc8c3a4 afbf1620 bddb83b1 f5dec6dd a20120c2 e7f8d8  
 90 f0cbc01 209ef581 908fd7bd bef70120 c0c688a7 fed0b3c1 3920a1ce fedfe18e a0e2bc01 20d286fb 98  
 a2dfbd 970120f5 9a87c58d 92ced72d 20849ea7 c3be8fe4 db1620ed b4d3d3cb fee8dd1d 20c1cf86 ddb6ed9  
 8 f14b20e4 fbcc81c1 8d9d955e 2085dde5 94bcb799 fdb40120 fee8cccf 8a8df480 2c20a182 e5f2e282 81b  
 80d20 848ad3ec 8ddcd9f 252094ac f5959abb b6c28501 20e7ae92 eee6f1ea b6c90120 b7dbd39a f0f385e5  
 81012080 cf84abe7 bd9cb666 20a0af89 f7f0d4b3 edc10120 868099e5 d8a7b3ec d801f805 b1908001 8a060  
 608 0110bd90 589a0605 0a034348 4e1a002a a101d201 9d010a97 01415041 52386e75 42734548 57666268 4  
 d635f37 71627344 5a697352 68544f33 65745849 526a4b42 327a7673 516c396f 5f41764c 58647245 733169  
 39 31417663 574a6a4a 646d4254 43613245 5a47765a 78677052 67525547 50535961 6b4a6166 73367067 4e  
 38796a 746e3658 7354466b 50567865 6c5f474b 70306b65 33676245 582d6738 437a3057 72776d5f 6a66673  
 5 46376a59 33523530 10d8044a e0020add 020a026d 7312d602 6e4b496e 4f44696b 50625735 6e5a4437 2d6  
 f7256 7773434e 44566759 536b6148 36454758 776d5f50 6c414b4b 47535659 51564854 6377335f 58655049  
 72456278 42545272 6d563355 55485457 462d4d52 4c446d64 35493950 67377631 5561336f 324d6a2d 42376  
 242 6e457a4b 5064666e 46344471 4f585a77 41695f30 5a512d76 79556348 526b586d 7444484f 38625166 7  
 4465452 56746c79 596f6a75 74554439 634d4b66 4723741 2d416675 7558304c 755f7a78 2d35724e 6a4c63  
 72 306a4a6b 31705266 6c4c794e 7a7a4262 5f782d71 5f6b3442 4b367438 346f6f6c 5779624c 43567463 6a  
 536774 6a7a4d34 50485139 33307448 7752795f 4a34466a 31525f7a 4b696b4a 59507a6c 544a5051 4e52504  
 9 6b43716e 766d7339 4b4e3875 4e704839 7a775130 6d756746 75564e50 4f476168 4f436255 62585077 4d4  
 3626d 5f667878 335f5935 54427631 68346e7a 6e613850 6e436c70 6367620a 0a084341 45534167 674312f7  
 02414d75 79323937 6d4f7158 3242787a 4f724b79 676b4973 59644448 6b4c6947 5f417972 53305975 6334  
 486e 6c73596d 6a373557 42484c62 484e374a 5f496744 48647862 476a4452 4e326c71 46395476 35444456  
 57514a69 35437a38 5553525f 7a524a53 7345392d 4e50594b 51353671 5f305776 6e4f646f 724e326b 31414  
 346 774c3545 6654584e 69767268 634e4e30 31687965 33766168 524a7833 774d364b 4a465534 5f4c6364 6

```
b647863 3363764a 6b537478 33302d63 5a59716d 746a6451 7a7a4d6c 6b4f5771 664b5a79 5a734e36 56526e
2d 612d5a67 64623370 7137726a 62414c55 486e7839 5f56686d 42304362 58344a6e 44616f68 4f644b4d 71
306278 54594f72 46394b73 3274704a 7964454f 3030534f 496a7270 705f4563 53693578 62434e65 64525147
58393667 6f414635 63443777 766b4146 46544475 3166414a 506e484e 46705335 68375f48 72553730 596d
446d 7a717647 724c564c 76347a45 30625f62 766c7043 76516853 43443731 44496c39 32356341 18a0d119
20012a48 0a462097 032889e5 18300038 03400048 00580062 2d766964 656f5f66 6f726d61 743d3232 26736
46b 763d692e 31372e30 38266f75 74707574 3d786d6c 5f766173 7432e801 00e80203 32104b4b 61304879 4
471544d 30533961 3641>
```



- 解析

```
po [objc_getClass("YTIAdBreakRequest") parseFromData: newHttpBodyData]
```

- 输出：json格式的Protobuf数据

```
(lldb) po [objc_getClass("YTIAdBreakRequest") parseFromData: newHttpBodyData]
2022-10-17 11:43:58.552972+0800 YouTube[21038:2204921] hook_ youtubeReqResp.xm YTIInnerTubeCont
ext$descriptor curDesc=<GPBDescriptor 0x2803e7a40>
2022-10-17 11:43:58.553697+0800 YouTube[21038:2204921] hook_ youtubeReqResp.xm YTIInnerTubeCont
ext$descriptor curDesc=<GPBDescriptor 0x2803e7a40>
2022-10-17 11:43:58.713094+0800 YouTube[21038:2204921] hook_ youtubeReqResp.xm YTIInnerTubeCont
ext$descriptor curDesc=<GPBDescriptor 0x2803e7a40>
<YTIAdBreakRequest 0x286120d20>: {
    context {
        client {
            hl: "zh-CN"
            gl: "CN"
            carrier_geo: "CN"
            device_make: "Apple"
            device_model: "iPhone9,1"
            client_name: IOS
            client_version: "17.08.2"
            os_name: "iOS"
            os_version: "13.3.1.17D50"
            screen_width_points: 375
            screen_height_points: 667
            screen_pixel_density: 2
            client_form_factor: SMALL_FORM_FACTOR
            window_width_points: 375
            window_height_points: 667
            connection_type: CONN_WIFI
            config_info {
                cold_config_data: "CMaZqJoGEOrKrgUQqM-oFxDUg64FEPeIrgUQ4rmuBRDS3K0FE0qdrQUQmcauBRDFmq
4FEMaFrQUQ19CuBRCD0K4FEIjprQUQuIuuBRDt1q0FEKXvrQUQ2butBRCGoa4FEOBnrgUQ1M-uBRDrma0FEIz
yruQOp52uBRDXka0FEIjUrQUQxb2uBRCCta4FEOTKrgUQgcWtBRDfzq4FEMexrQUQ2LyuBRDX9a0FENvKrgUQ78iu
BRDq16wFEImxrgUQgb1uBRD7zq4FEJaarQUQmr6uBRD11q4FEIGGrgUQ0_GtBRCboK4FEKH9rQUQoLmuBRDN4K0F
EJPQrgUQnv2tBRDr5K0FEIa1rgUQ6rqtBRCGy64FEOTNrgUQ9MetBRC1sK4FGjJBS3B1ZGh5TTV0VKn1QW92c01T
UmRuajd1bF80RmhENmpMwnZUY2E5RXRSOGhGVkRRQSIyQuTwZW RoeU01dFZDdUFvdnNNU1Jkbmo3Ymx
fNEZoRDZqTFp2VGNhOUV0Ujh0R1ZEUEqGENBTVNEQTBNk0tcEFoVuQ4dDNYREE9PQ%3D%3D"
                cold_hash_data: "CJPwsSpoGEhMyOTA4MTIwNDgzNTkxNTk3NTU0GMaZqJoGKJnGrgUozeCtBSiGta4FKOTK
rgUojPKtBSj0x60FKKwrgUo5s2uBSjZu60FKJugrgUooLmuBSjrm
a0FKJTPrgUog9CuBS1avq4FKL1LrgUo19CuBSiI6a0
FKN18rgUo6p2tBSj1lq4FKNSDrgUo6pesBS1BuK4FKOrKrgUo941uBS1oz6gXKImxrgUoxb2uBSjkza4FKNeRrQ
UohsuuBSinna4FKJPQrgUo6rqtBSiCta4FKIHFrQuoof2tBSiG
oa4FKNPxrQuo7YqtBSie_a0FKJjUrQuolpqtBSjGha0FKIGGrgUop
            }
        }
    }
}
```

```

e-tBSjS3K0FKNf1rQUox7GtBSjFmq4FKPvOrgUo78iuBSjbyq4FKN_OrgUo4rmuBSjr5K0FMjjBS3B1ZGh5TTV0Vkn1QW92
c01TUmRuajdibF80RmhENmpMWhZUY2E5RXRSOGhGVkRRQToyQUtzwWRoeU01dFZDdUFVdnNNU1Jkbmo3YmxNEzoRDZqTFp
2VGNhOUV0Ujh0R1ZEUUFCGENBTVNEQTBNk0tcEFoVUQ4dDNYREE9PQ%3D%3D"
    hot_hash_data: "CJPwspoGEhQxNDIwOTM5NzAxNjAzODg3NTExMRiT8LKaBiiyiP4SKLzK_RIo34z-EiiAg
v4SK02E_hIo89_9EijLrf0SKL6J_hIo1YL-EiiSiv4SKIrg_RIo3o3-E1jjhP4SKKTE_RIon4f-EiiA8_0SKMH-_RIopd9
E11U5PwSKOK__RIogYL9EiiPjP4SKNPe_RIo1PD9EijKh_4SKMWO_hIo5YD-E1jj-f0SKNGf_RIo54l-E1j53v0SKO1C_hI
o3D9EijhrP0SKNLL_Rionv_8Eijbk_0SKNyK_hIowIP-Eii93v0SKP2s_RIo-y-z9EijbgP4SKKf8_RIoqrT9EijGsv0SKK
mq_Riomcb9Ei18vv0SKJvM_RIo-IV-Eij0wP0SKIEc_hIoj4X-Eii59fwSKNFm_RIoaz9EijK2P0SMjJBs3B1ZGh5TTV0V
kn1QW92c01TUmRuajdibF80RmhENmpMWhZUY2E5RXRSOGhGVkRRQToyQUtzwWRoeU01dFZDdUFVdnNNU1Jkbmo3YmxNEzo
RDZqTFp2VGNhOUV0Ujh0R1ZEUUFCHENBTVNFTBCb3RmNkZad3BGUURkejhJTW5jd0U%3D"
}

screen_density_float 2
utc_offset_minutes 480
user_interface_theme USER_INTERFACE_THEME_LIGHT
time_zone "Asia/Shanghai"
eml_template_context " \345\211\377\317\331\253\214\373\222\001 \203\370\325\224\370\3
30\256\322V \325\301\314\367\223\207\327\373U \276\204\207\214\212\376\360\304\347\001 \331\335
\220\314\204\217\363\200\001 \332\336\224\374\252\254\271\220Y \306\272\225\227\264\371\311
\263a \206\362\253\252\224\317\256\355 \305\305\224\210\315\337\377\207\232\001 \342\241\300\2
60\346\337\227\362/ \235\241\315\245\210\324\374\360~ \375\216\303\242\370\305\303\320\003 \331
\250\231\346\255\224\333\371\037 \225\245\366\315\203\363\377\232\024 \334\355\231\374\342\317\
336\223\332\001 \264\230\275\310\303\244\257\277\026 \275\333\203\261\365\336\306\335\242\001 \
302\347\370\330\220\360\313\312\001 \236\365\201\220\217\327\275\276\367\001 \300\306\210\247\3
76\320\263\3019 \241\316\376\337\341\216\240\342\274\001 \322\206\373\230\242\337\275\227\001 \
365\232\207\305\215\222\316\327- \204\236\247\303\276\217\344\333\026 \355\264\323\323\313\376\
350\335\035 \301\317\206\335\266\355\230\361K \344\373\314\201\301\215\235\225^ \205\335\345\22
4\274\267\231\375\264\001 \376\350\314\317\212\215\364\200, \241\202\345\362\342\202\201\270\r
\204\212\323\354\215\334\334\237% \224\254\365\225\232\273\266\302\205\001 \347\256\222\356\346
\361\352\266\311\001 \267\333\323\232\360\363\205\345\201\001 \200\317\204\253\347\275\234\266f
\240\257\211\367\360\324\263\355\301\001 \206\200\231\345\330\247\263\354\330\001"
memory_total_kbytes 2099249
notification_permission_info {
    notifications_setting: NOTIFICATIONS_SETTING_ENABLED
    last_device_opt_in_change_time_ago_sec 1443901
}
client_store_info {
    ios_store_country "CHN"
}
}
user {
}
request {
    consistency_token_jars {
        encrypted_token_jar_contents "APAR8nuBsEHWfbhMc_7qbsDZisRhT03etXIRjKB2zvsQl9o_AvLXdr
Es1191AvcWJjJdmBTCa2EZGvZxgpRgRUGPSYakJafs6pgN8yjtn6XsTFkPVxel_GKp0ke3gbEX-g8Cz0Wrwm_jfg5F7jY3R
50"
        expiration_seconds 600
    }
}
ad_signals_info {
    params {
        key: "ms"
        value: "nKInOD1kPbW5nZD7-orVwsCNDVgYSkaH6EGXwm_P1AKKGSVYQVHTcw3_XePIrEbxBTRrmV3UUHTWF
-MRLDmd5I9Pg7v1Ua3o2Mj-B7bBnEzKPdfnF4DqOXZwAi_0ZQ-vyUcHRkXmtDH08bQftFTRVtlyYojutUD9cMKfGr7A-Afu
uX0Lu_zx-5rNjLcr0jJk1pRf1LyNzBb_x-q_k4BK6t84oo1WybLCVtcjSgtjzM4PHQ930tHwRy_J4Fj1R_zKikJYPz1TJP
QRPIkCqnvms9KN8uNpH9zwQ0mugFuVNPOGAhOCbUbXPwMCbm_fxx3_Y5TBv1h4nzna8PnC1pcg"
}

```

```
        }
    }

    active_players {
        player_context_params: "CAESAaggC"
    }
}

params: "AMuy297mOqX2BxzOrKygkIsYdDHkL1G_AyrS0Yuc4HnlsYmj75WBHLbHN7J_IgDHdxBGjDRN2lqF9Tv5DD
VWQJ15Cz8USR_zRJSsE9-NPYKQ56q_0Wvn0dorN2k1ACFwL5EfTXNivrhcNN01hye3vahRJx3wM6KJFU4_LcdKdxc3cvJks
tx30-cZYqmtjdQzzM1k0WqfKZyZsN6VRn-a-Zgdb3pq7rjbALUHnx9_VhmB0CbX4JnDaoh0dKMq0bxTYOrF9Ks2tpJydE00
0SOIjrpp_EcSi5xbCNedRQGX96goAF5cD7wvkAFFTDu1fAJPnHNFpS5h7_HrU70YmDmzqvGrLVLv4zE0b_bvlpCvQhSCD71
DI1925cA"
break_position_ms: 420000
break_index 1
override_playback_context {
    content_playback_context {
        time_since_last_ad_seconds: 407
        last_milliseconds 406153
        autoplays_since_last_ad: 0
        conn: 3
        vis: 0
        fling: false
        autoplay false
        adsense_client_params: "video_format=22&sdkv=1.17.08&output=xml_vast2"
        autonav false
        autonav_state: STATE_OFF
    }
}
client_playback_nonce: "KKa0HyDqTM0S9a6A"
}
```

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:  
2022-11-04 15:33:59

## YTINnerTubeContext

TODO:

- 【未解决】研究YouTube逻辑：从OnesieRequestProto子属性类型YTINnerTubeContext找到生成data的机制

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：

2022-11-04 15:46:32

## YTIClientInfo

- 【未解决】研究YouTube逻辑：搞懂protobuf类YTIClientInfo的字段定义
- 

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：  
2022-11-04 14:52:14

## lite的C++类

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：  
2022-11-04 15:44:48

# OnesieRequestProto

## TODO:

- 【未解决】研究YouTube逻辑：lite版protobuf类OnesieRequestProto如何搞清楚属性字段定义
  - 【未解决】研究YouTube逻辑：OnesieRequestProto相关

## OnesieRequestProto的vtable定义

经过逆向，目前的理解是：

## IDA中改名：

## 核心定义：

```
00000004F7BEE8 ; `vtable for`video_streaming::OnesieRequestProto
__const:0000000004F7BEE8 __ZTVN15video_streaming18OnesieRequestProtoE DCQ 0 ; offset to this
__const:0000000004F7BEF0 DCQ __ZTIN15video_streaming18OnesieRequestProtoE ; `ty
peinfo for`video_streaming::OnesieRequestProto
__const:0000000004F7BEF8 MessageLite_commonDctor_4F7BEF8 DCQ OnesieRequestProto_commonDctor_32D
3230
__const:0000000004F7BEF8 ; DATA XREF: requestConstructo
r_32D2E8C Cto
__const:0000000004F7BEF8 ; requestConstructor_32D2E8C 10
to ...
const:0000000004F7BF00 DCQ OnesieRequestProto_dtor_32D3234
```

```

__const 000000004F7BF08          DCQ OnesieRequestProto_GetTypeName_32D3ED8
__const 000000004F7BF10         DCQ OnesieRequestProto_NewArena_32D3FA0
__const 000000004F7BF18         DCQ OnesieRequestProto_Clear_32D3248
__const 000000004F7BF20         DCQ OnesieRequestProto_IsInitialized_32D3EA8
__const 000000004F7BF28         DCQ MessageLite_InitializationErrorString_3891244
__const 000000004F7BF30         DCQ OnesieRequestProto_CheckTypeAndMergeFrom_32D3EA4
__const 000000004F7BF38         DCQ OnesieRequestProto_ByteSizeLong_32D3BAC
__const 000000004F7BF40         DCQ __ZNK7youtube8elements6Entity13GetCachedSizeEv ; y
outube::elements::Entity::GetCachedSize(void)
__const 000000004F7BF48         DCQ OnesieRequestProto__InternalParse_32D338C
__const 000000004F7BF50         DCQ __ZN6proto28internal124GeneratedExtensionFinderD1Ev
; proto2::internal::GeneratedExtensionFinder::GeneratedExtensionFinder()
__const 000000004F7BF58         DCQ OnesieRequestProto__InternalSerialize_32D373C
__const 000000004F7BF60 ; public video_streaming::OnesieRequestProto :

```

OnesieRequestProto的vtable定义：

- vtable OnesieRequestProto
  - +0x08 = OnesieRequestProto\_typeinfo
  - +0x10 = MessageLite\_commonDctor\_4F7BEF8 = OnesieRequestProto\_commonDctor\_32D3230
    - 某种 deconstructor ?
  - +0x18 = OnesieRequestProto\_dtor\_32D3234
    - virtual ~MessageLite() = default;
  - +0x20 = OnesieRequestProto\_GetTypeName\_32D3ED8
    - virtual std::string GetTypeName() const = 0;
  - +0x28 = OnesieRequestProto\_NewArena\_32D3FA0
    - virtual MessageLite New(Arena arena) const = 0;
  - +0x30 = OnesieRequestProto\_Clear\_32D3248
    - virtual void Clear() = 0;
  - +0x38 = OnesieRequestProto\_IsInitialized\_32D3EA8
    - virtual bool IsInitialized() const = 0;
  - +0x40 = MessageLite\_InitializationErrorString\_3891244
    - virtual std::string InitializationErrorString() const;
  - +0x48 = OnesieRequestProto\_CheckTypeAndMergeFrom\_32D3EA4
    - virtual void CheckTypeAndMergeFrom(const MessageLite& other) = 0;
  - +0x50 = OnesieRequestProto\_ByteSizeLong\_32D3BAC
    - virtual size\_t ByteSizeLong() const = 0;
  - +0x58 = youtube::elements::Entity::GetCachedSize(void)
    - virtual int GetCachedSize() const = 0;
  - +0x60 = OnesieRequestProto\_\_InternalParse\_32D338C
    - virtual const char \_InternalParse(const char /ptr/, internal::ParseContext /ctx\*)
  - +0x68 = proto2::internal::GeneratedExtensionFinder::~GeneratedExtensionFinder()
    - virtual void OnDemandRegisterArenaDtor(Arena /arena\*) {}
  - +0x70 = OnesieRequestProto\_\_InternalSerialize\_32D373C
    - virtual uint8\_t \_InternalSerialize(uint8\_t ptr, io::EpsCopyOutputStream\* stream) const = 0;



## 附录

下面列出相关参考资料。

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：  
2022-11-04 14:44:04

## 参考资料

- 【整理】Protocol Buffers即protobuf总结
- 【未解决】研究YouTube逻辑：OnesieRequestProto类的具体vtable函数实现
- 【已解决】研究YouTube逻辑：从NSData直接解析出protobuf的类YTIAdBreakRequest
- 
- [protobuf/src/google/protobuf/message\\_lite.h - chromium/src/third\\_party - Git at Google \(googlesource.com\)](#)
- [message\\_lite.h | Protocol Buffers | Google Developers](#)
- [third\\_party/protobuf/src/google/protobuf/message\\_lite.cc - chromium/src - Git at Google \(googlesource.com\)](#)
- [PB协议报错 it is missing required fields: \(cannot determine missing fields for lite message\) \\_军说网事的博客-CSDN博客](#)
- [examples - external/github.com/google/protobuf - Git at Google \(googlesource.com\)](#)
- 

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：  
2022-11-04 15:38:48