目录

[OBD结构Protobuf定义 1](#_Toc425794344)

[1.  概述 1](#_Toc425794345)

[1.1 目的 1](#_Toc425794346)

[1.2 适用范围 1](#_Toc425794347)

[2.  Protobuf简介 1](#_Toc425794348)

[2.1如何工作 1](#_Toc425794349)

[2.2字段指定规则 2](#_Toc425794350)

[2.3变量类型 2](#_Toc425794351)

[3.  数据结构 4](#_Toc425794352)

[3.1程序包定义 4](#_Toc425794353)

[3.2 OBD消息结构定义 4](#_Toc425794354)

[4.  附录 13](#_Toc425794355)

[4.1设备生产厂商标识 13](#_Toc425794356)

[4.2消息类型定义 13](#_Toc425794357)

# OBD结构Protobuf定义

## 1.  概述

### 1.1 目的

在数据跨系统传输情况，需要确保所有系统对同一数据结构/对象使用相同的序列化和反序列化机制；OBD系统采用Protocol Buffer作为数据序列/反序列化方式。

### 1.2 适用范围

适用于公车管理系统开发人员。

## 2.  Protobuf简介

※ Protobuf 是Google 的一种数据交换的格式，它独立于语言，独立于平台。由于它是一种二进制的格式，比使用 xml 进行数据交换要快的许多。可以把它用于分布式应用之间的数据通信或者异构环境下的数据交换。作为一种效率和兼容性都很优秀的二进制数据传输格式，可以用于诸如网络传输、配置文件、数据存储等诸多领域。

※ 项目中用到Protobuf进行数据序列化，好处在于后期程序扩展性非常好，只需要改动proto的定义就可以保持兼容，非常的灵活方便。

### 2.1如何工作

※ 首先需要创建一个 .proto 文件，在文件中定义你需要做串行化的数据结构，在每个数据结构里定义需要序列化的成员对象。

※ 下面举例说明User对象的定义规范，文件名称为user.proto该文件的内容大致如下：

message Order  
{  
  required int32 id = 1;  
  required string name= 2;  
  required int32 age= 3;  
  optional string address= 4;  
}

※ 通过Protobuf 工具从user.proto文件生成（Java，C#，Python）的源代码，引入项目中就可以使用。

### 2.2字段指定规则

所指定的消息字段修饰符必须是如下之一：

* required：一个格式良好的消息一定要含有1个这种字段。表示该值是必须要设置的；
* optional：消息格式中该字段可以有0个或1个值（不超过1个）。
* repeated：在一个格式良好的消息中，这种字段可以重复任意多次（包括0次）。重复的值的顺序会被保留。

### 2.3变量类型

|  |  |  |  |
| --- | --- | --- | --- |
| .proto类型 | Java 类型 | C++类型 | 备注 |
| double | double | double |  |
| float | float | float |  |
| int32 | Integer | int32 | 使用可变长编码方式。编码负数时不够高效——如果你的字段可能含有负数，那么请使用sint32。 |
| int64 | long | int64 | 使用可变长编码方式。编码负数时不够高效——如果你的字段可能含有负数，那么请使用sint64。 |
| uint32 | Integer | uint32 | Uses variable-length encoding. |
| uint64 | long[1] | uint64 | Uses variable-length encoding. |
| sint32 | Integer | int32 | 使用可变长编码方式。有符号的整型值。编码时比通常的int32高效。 |
| sint64 | long | int64 | 使用可变长编码方式。有符号的整型值。编码时比通常的int64高效。 |
| fixed32 | Integer | uint32 | 总是4个字节。如果数值总是比228大的话，这个类型会比uint32高效。 |
| fixed64 | long[1] | uint64 | 总是8个字节。如果数值总是比256大的话，这个类型会比uint64高效。 |
| sfixed32 | Integer | int32 | 总是4个字节。 |
| sfixed64 | long | int64 | 总是8个字节。 |
| bool | Boolean | bool |  |
| string | String | string | 一个字符串必须是UTF-8编码或者7-bit ASCII编码的文本。 |
| bytes | ByteString | string | 可能包含任意顺序的字节数据。 |

## 3.  数据结构

对OBD数据的Protobuf结构定义

### 3.1程序包定义

※ 程序包名称：com.wttech.obd.persistence

※ 对于Java系统，程序包名是Package名，对于C#系统，是命名空间名称（其他语言可忽略该项）。

### 3.2 OBD消息结构定义

1. **消息OBDMessage**

message OBDMessage {

required int32 obdManufacture = 1;

required string obdId = 2;

required int32 messageType = 3;

optional GpsMessage gpsMessage = 4;

optional ConMessage conMessage = 5;

optional GSensorMessage gSensorMessage = 6;

optional ConTypeMessage conTypeMessage = 7;

optional SnapShotMessage snapShotMessage = 8;

optional HVFaultMessage hvFaultMessage = 9;

optional CVFaultMessage cvFaultMessage = 10;

optional AlarmMessage alarmMessage = 11;

optional BaseStationMessage baseStationMessage = 12;

optional TimingGpsMessage timingGpsMessage = 13;

}

obdManufacture：设备生产厂商标识，参考附录1

obdId：OBD设备唯一标识

messageType：消息类型定义，参考附录2

gpsMessage：Gps消息内容

conMessage：工况消息内容

gSensorMessage：GSensor数据

conTypeMessage：支持数据流类型数据

snapShotMessage：快照数据

hvFaultMessage：乘用车故障数据

cvFaultMessage：商用车故障数据

alarmMessage：警情数据

baseStationMessage：基站数据

timingGpsMessage：睡眠定时上传数据

1. **ObdDate日期结构**

message ObdDate{

required int32 year=1;

required int32 month= 2;

required int32 day= 3;

}

year：年

month：月

day：日

1. **ObdTime时间结构**

message ObdTime{

required int32 hour= 1;

required int32 minute= 2;

required int32 second= 3;

}

hour：时

minute：分

second：秒

1. **Gps结构**

message GpsItem{

required ObdDate date=1;

required ObdTime time=2;

required double lat=3;

required double lon=4;

required int32 speed=5;

required int32 dir=6;

required int32 valFlag=7;

}

date：GPS日期

time：GPS时间

lat：纬度

lon：经度

speed：速度

dir：方向

valfalg：定位标志

1. **Rpm结构**

message RpmItem {

required int32 rpm = 1;

}

rpm：发送机转速

1. **VStat结构**

message VStat{

required int32 S0= 1;

required int32 S1= 2;

required int32 S2= 3;

required int32 S3= 4;

}

1. **StatData结构**

message StatData{

required int32 lastAcconTime= 1;

required int32 utcTime= 2;

required int32 totalTripMileage = 3;

required int32 currentTripMileage = 4;

required int32 totalFuel = 5;

required int32 currentFuel= 6;

required VStat stat= 7;

required int64 reserve = 8;

}

lastAcconTime：最近ACC点火时间

utcTime：设备时间

totalTripMileage：累计里程

currentTripMileage：当前里程

totalFuel：累计油耗

currentFuel：当前油耗

stat：状态包

reserve：预留

1. **GSensorData结构**

message GSensorData{

required int32 x= 1;

required int32 y= 2;

required int32 z= 3;

}

x：x轴加速度

y：y轴加速度

z：z轴加速度

1. **AlarmData结构**

message AlarmData{

required int32 newAlarmFlag= 1;

required int32 alarmType= 2;

required int32 alarmDesc= 3;

required int32 alarmThreshold= 4;

}

newAlarmFlag：新警情标识

alarmType：警情类型

alarmDesc：当前值

alarmThreshold：警情阈值

1. **GpsMessage 数据**

message GpsMessage {

required int32 flag = 1;

required StatData statData = 2;

repeated GpsItem gpsData = 3;

repeated RpmItem rpmData = 4;

}

flag：GPS数据标识

statData：统计数据包

gpsItem ：1个包的GpsItem 结构

rpmData :RPM数据

1. **ConMessage工控数据**

message ConMessage {

required StatData statData = 1;

required int32 collectInterval= 2;

repeated int32 conTypeArray=3;

required int32 conGroupCount =4;

required int32 conGroupSize =5;

optional bytes conData =6;

}

statData：统计数据包

collectInterval：采集间隔

conTypeArray：工况类型数组

conGroupCount：工况数据包数

conGroupSize：每包工况数据包数

conData：工况数据

1. **GSensorMessage数据**

message GSensorMessage {

required StatData statData = 1;

required int32 collectInterval= 2;

repeated GSensorData gSensorData=3;

}

statData：统计数据包

collectInterval：采集间隔

gSensorData：GSensor数组

1. **ConTypeMessage数据**

message ConTypeMessage {

required StatData statData = 1;

repeated int32 conTypeArray= 2;

}

statData：统计数据包

dataTypeArray：数据流类型数组

1. **SnapShotMessage数据**

message SnapShotMessage {

required StatData statData = 1;

required int32 frozenFlag= 2;

repeated int32 dataTypeArray=3;

optional bytes dataContent=4;

}

statData：统计数据包

frozenFlag：冻结帧标志

dataTypeArray：数据流类型数组

dataContent：数据流值数组

1. **HVFaultMessage数据**

message HVFaultMessage {

required StatData statData = 1;

required int32 faultFlag= 2;

repeated int32 faultTypeArray=3;

}

statData：统计数据包

faultFlag：故障标识

faultTypeArray：故障数组

1. **CVFaultMessage数据**

message CVFaultMessage {

required StatData statData = 1;

required int32 faultFlag= 2;

repeated int32 faultTypeArray=3;

}

statData：统计数据包

faultFlag：故障标识

faultTypeArray：故障数组

1. **AlarmMessage数据**

message AlarmMessage {

required int32 alarmNum= 1;

required StatData statData = 2;

repeated GpsItem gpsData = 3;

repeated AlarmData alarmData=4;

}

alarmNum：警情编号

statData：统计数据包

gpsData：GPS数据

alarmData：警情消息

1. **BaseStationMessage数据**

message BaseStationMessage{

required StatData statData = 1;

required int32 locationAreaCode= 2;

required int32 cellId= 3;

}

statData：统计数据包

locationAreaCode：基站位置编号

cellId：基站小区ID编号

1. **TimingGpsMessage数据**

message TimingGpsMessage {

required int32 utcTime= 1;

required GpsItem gpsItem= 2;

}

utcTime ：设备当前UTC时间

gpsItem ：1个包的GpsItem 结构

## 4.  附录

### 4.1设备生产厂商标识

|  |  |  |
| --- | --- | --- |
| **常量** | **设备厂家** | **描述** |
| 1 | 航天 | 航天的OBD设备 |

### 4.2消息类型定义

|  |  |  |
| --- | --- | --- |
| 常量 | 消息类型 | 描述 |
| 0x0001 | gpsMessage | GPS消息 |
| 0x0002 | conMessage | 工况数据 |
| 0x0004 | gSensorMessage | GSensor数据 |
| 0x0008 | conTypeMessage | 支持数据流数据 |
| 0x0010 | snapShotMessage | 快照数据 |
| 0x0020 | hvFaultMessage | 乘用车故障数据 |
| 0x0040 | cvFaultMessage | 商用车故障数据 |
| 0x0080 | alarmMessage | 警情数据 |
| 0x0100 | baseStationMessage | 基站定位数据 |
| 0x0200 | timingGpsMessage | 睡眠定时上传数据 |