# OBD 通信系统设计概要

## 1.  引言

### 1.1 项目概述

OBD 通信系统是公车管理系统的一部分，负责同终端设备的通信，起到桥梁作用，实现跨区域同时运维车辆的目的。

### 1.2 适用范围

适用于公车管理通信系统开发人员。

### 1.3 术语定义

**API：**（Application Programming Interface,应用程序编程接口）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需理解内部工作机制的细节。

**Web API：**是网络应用程序接口。

**HTTP**：超文本传输协议（HTTP，Hyper Text Transfer Protocol)是互联网上应用最为广泛的一种网络协议。所有的WWW文件都必须遵守这个标准。

**JavaScript：**一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。为浏览器的一部分，广泛用于客户端的脚本语言。

**OBD：**OBD是英文On-Board Diagnostic的缩写，中文翻译为“车载诊断系统”。这个系统随时监控发动机的运行状况和尾气后处理系统的工作状态，一旦发现有可能引起排放超标的情况，会马上发出警示。

**序列化**：(Serialization)将对象的状态信息转换为可以存储或传输的形式的过程。在序列化期间，对象将其当前状态写入到临时或持久性存储区。以后，可以通过从存储区中读取或反序列化对象的状态，重新创建该对象。

**队列**：是一种特殊的线性表，特殊之处在于它只允许在表的前端（front）进行删除操作，而在表的后端（rear）进行插入操作和堆栈一样，队列是一种受限制的线性表。

**缓存**：是数据交换的缓冲区（称作Cache），当系统要读取数据时，会首先从缓存中查找需要的数据，如果找到了则直接复制，找不到的话则从其他介质中查找。缓存工作的原则，就是“引用的局部性”，这可以分为时间局部性和空间局部性。

**AJAX**：AJAX即“Asynchronous JavaScript And XML”（异步JavaScript和XML），是指一种创建交互式网页应用的网页开发技术。

**Servlet**：是 Java 编程语言中的一个类，它被用来扩展服务器的性能，服务器上驻留着可以通过“请求-响应”编程模型来访问的应用程序。虽然 Servlet 可以对任何类型的请求产生响应，但通常只用来扩展 Web 服务器的应用程序。

**JSON：**JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于ECMA Script的一个子集。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯。这些特性使JSON成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成（网络传输速率）。

**Protocol Buffer：**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进行数据交换速度快。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。

### 1.4 参考文献

《计算机文化（第五版）》June jamrich Parsons，Dan Oja．

《计算机信息处理》Steven L. Mandell，Sachi Sakthivel

《计算机网络（第四版）》谢希仁

## 2.  方案设计

## 2.1 系统设计

系统整体分为五个主要部分

1. OBD登录服务器
2. OBD通信服务器
3. OBD设备Web管理页面
4. OBD设备Web API接口
5. OBD模拟测试工具

## 2.2.功能设计

## 2.2.1 OBD登录服务器

OBD 登录服务器模块划分如下：

1. 网络通信：负责管理客户端的网络接入和数据的读写功能。
2. 状态维护模块：检查客户端连接状态，移除失效网络连接。
3. 协议模块：实现所有OBD命令的编解码。
4. 序列化模块：通过Protocol Buffer对OBD命令进行序列化/反序列化。
5. 配置管理：通过配置文件、命令参数统一管理系统运行参数。
6. 缓存管理：使用REDIS维护缓存数据，维护登陆运行时信息。
7. 存储队列：将序列化后的数据保存到KAFKA队列供其他系统对接使用。
8. HTTP代理：实现HTTP代理功能，通过允许其他系统通过Web API对设备进行控制。

## 2.2.2 OBD通信服务器

OBD 通信服务器模块划分如下：

1. 网络通信：负责管理客户端的网络接入和数据的读写功能。
2. 状态维护：检查客户端连接状态，移除失效网络连接。
3. 协议模块：实现所有OBD命令的编解码。
4. 配置管理：通过配置文件、命令参数统一管理系统运行参数。
5. 缓存管理：使用REDIS维护缓存数据，维护登陆运行时信息。
6. 优先派发策略：根据每个服务器的运行时信息计算出空闲服务器供OBD客户端通信使用。

## 2.2.3 OBD设备管理页面

OBD 设备管理页面主要包括以下部分：

1. 设备联网状态查询
2. 获取设备位置信息
3. 清除终端设备故障信息
4. 恢复设备出厂设置
5. 终端发送信息
6. 获取终端设备参数
7. 设置终端设备参数

## 2.2.4 OBD Web API接口

OBD WEB API接口定义：

1. 查询终端参数接口
2. 设置终端参数接口
3. 恢复出厂设置接口
4. 获取设备的GPS信息接口
5. 清除故障信息接口
6. 文字信息发送接口

参见《OBD Web Api接口定义》

## 2.2.5 OBD模拟测试工具

OBD 模拟测试工具包括以下功能：

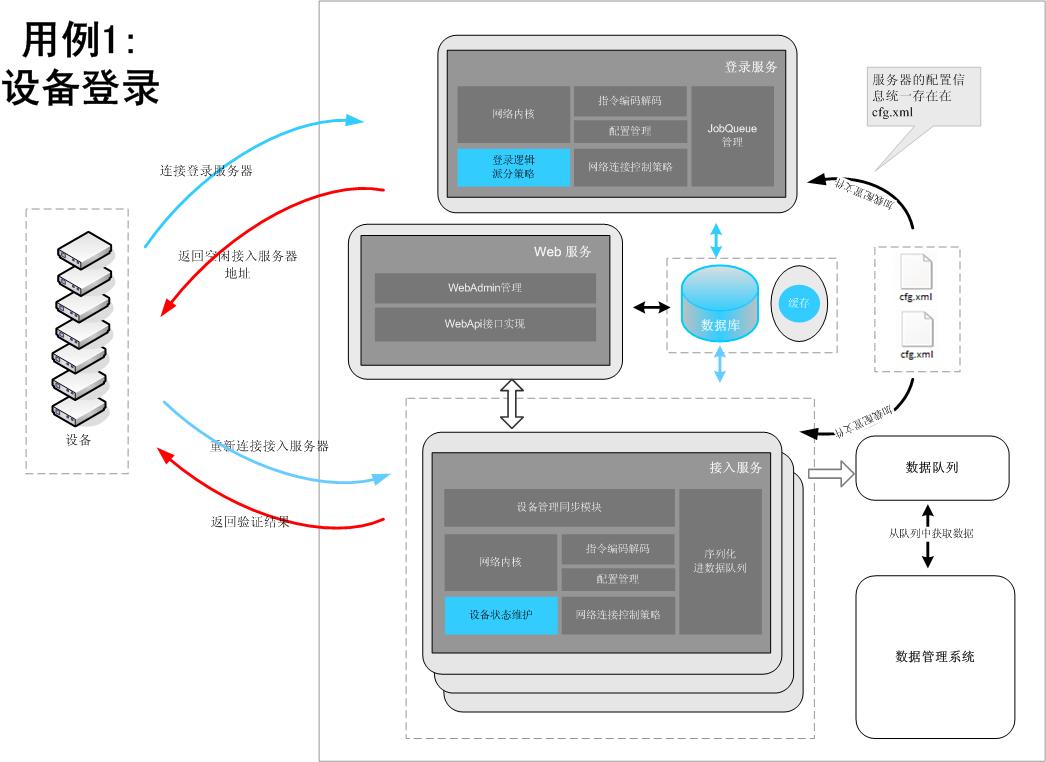
1. 模拟OBD设备报文发送
2. 模拟多个OBD设备在线连接
3. 模拟定时心跳机制
4. 配置设备报文参数

## 3.  用例分析

OBD 通信系统使用案例包括登录、通信和管理这三个主要案例，下面我们来重点介绍登录和管理案例。

### 3.1设备登录

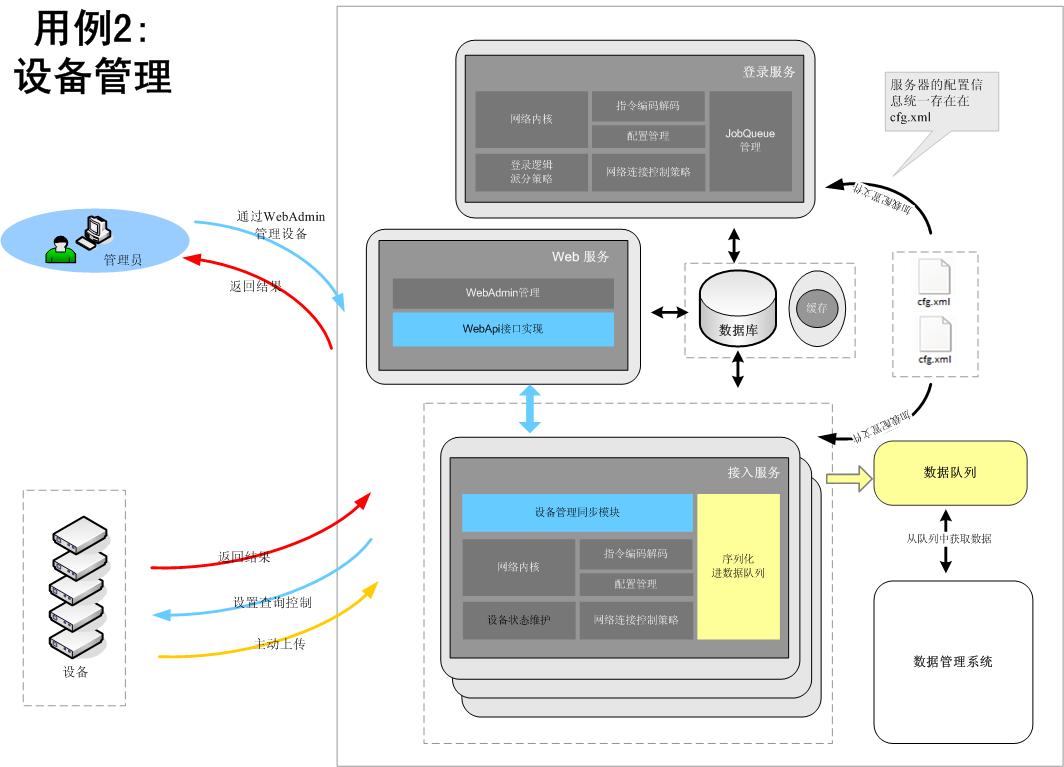
* 案例1：设备登录



1. 配置OBD设备的服务器地址指向为登录服务器地址。
2. 上电后OBD设备向登录服务程序发送登录请求报文。
3. 登录服务程序收到设备登录请求后，解码登录请求并校验，校验失败关闭链接。
4. 校验成功后，访问缓存系统中的服务器运行信息，并计算出空闲的通信服务器地址。
5. 编码登录回复命令，并下发给OBD设备。
6. 设备收到登录回复后，判断转接服务器地址字段是否为空，为空时登录过程结束。
7. 不为空时，保存通信服务器地址，并向通信服务器发送登录请求。
8. 通信服务程序收到设备登录请求后，解码登录请求并校验，校验失败关闭链接。
9. 校验成功后，编码登录回复命令，并下发给OBD设备。
10. OBD设备收到登录回复报文，登录过程结束。

### 3.2设备管理

* 案例2：设备管理



1. 管理员通过设备管理页面，选择OBD设备唯一编码。
2. 管理页面使用AJAX方式提交JSON编码的设备控制命令到Web服务器端。
3. 命令发送到Web服务器，服务器根据OBD设备唯一编码计算出该设备所在通信服务器地址。
4. 编码控制命令后通过Web API方式发送到通信服务器。
5. 通信服务器收到命令后解码并下发到设备终端，同时启动超时触发机制。
6. 如果触发超时，返回给Web服务器超时消息，Web服务器返回页面调用端JSON格式的失败命令。

如果成功，发送设备回复命令到Web服务器端。

1. Web服务器解码OBD回复命令并发送JSON编码的回复信息给管理员页面。
2. 页面通过JavaScript脚本填充数据到页面控件。

## 4.  缓存设计

OBD通信系统采用REDIS作为缓存介质，将保存所有OBD 设备链接信息和OBD Server的运行时信息，供后端系统对OBD设备和OBD Server的快速检索。

## 4.1服务程序运行时信息

记录通信服务器运行信息

1. 数据库KEY值

格式OBDSERVER\_[IP]:[PORT]

IP：服务程序地址

PORT：服务程序监听端口

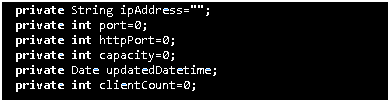
例如OBDSERVER\_127.0.0.1:8907

1. 数据库索引ID值

对于服务程序为固定值1df20ae0-0a6e-4b8e-ae03-92e1eb5947f1

1. 数据库VALUE值

格式：JSON格式的ObdCacheDataServer数据结构。



ipAddress：OBD Server服务器地址

port：OBD Server服务器端口

httpPort：本地监听的HTTP端口

capacity：OBD Server能容纳的最大客户端数量

updatedDatetime：OBD Server状态更新时间

clientCount：当前维护链接的OBD设备数量

### 4.2 OBD设备链接信息

记录OBD设备链接信息

1. 数据库KEY值

格式OBDSERVER\_[IP]:[PORT]

IP：服务程序地址

PORT：服务程序监听端口

例如OBDSERVER\_127.0.0.1:8907

1. 数据库索引ID值

OBD设备的OBD Number

1. 数据库VALUE值

OBD设备的Channel ID

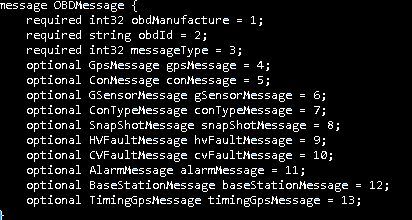
参见《OBD 缓存结构设计》

## 5.  OBD数据结构定义

OBD通信系统使用Protocol Buffer定义所有OBD命令格式，确保OBD命令在多个系统间传输时的对数据的序列化和反序列化一致。

### 5.1 OBD消息结构定义

1. 消息OBDMessage



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



year：年

month：月

day：日

1. ObdTime时间结构

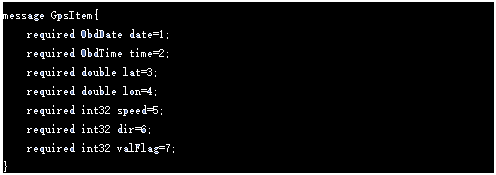


hour：时

minute：分

second：秒

1. Gps结构



date：GPS日期

time：GPS时间

lat：纬度

lon：经度

speed：速度

dir：方向

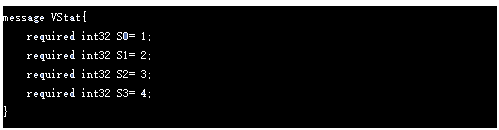
valfalg：定位标志

1. Rpm结构

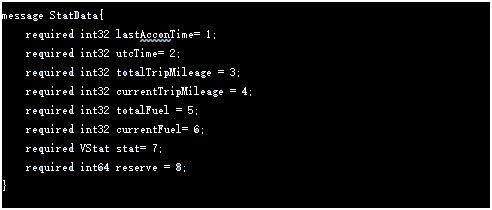


rpm：发送机转速

1. VStat结构



1. StatData结构

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

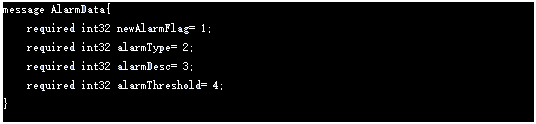


x：x轴加速度

y：y轴加速度

z：z轴加速度

1. AlarmData结构



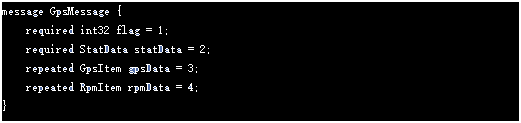
newAlarmFlag：新警情标识

alarmType：警情类型

alarmDesc：当前值

alarmThreshold：警情阈值

1. GpsMessage 数据



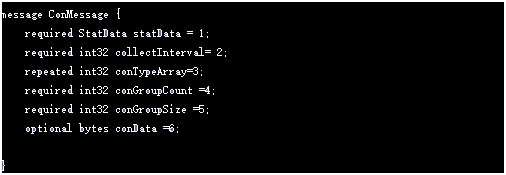
flag：GPS数据标识

statData：统计数据包

gpsItem ：1个包的GpsItem 结构

rpmData :RPM数据

1. ConMessage工控数据



statData：统计数据包

collectInterval：采集间隔

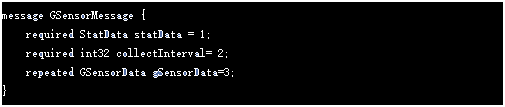
conTypeArray：工况类型数组

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



statData：统计数据包

collectInterval：采集间隔

gSensorData：GSensor数组

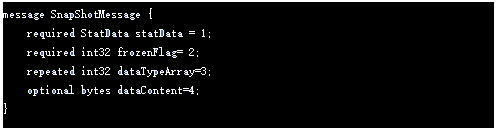
1. ConTypeMessage数据



statData：统计数据包

dataTypeArray：数据流类型数组

1. SnapShotMessage数据



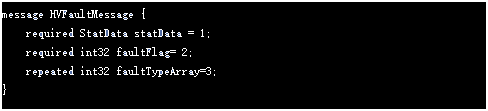
statData：统计数据包

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

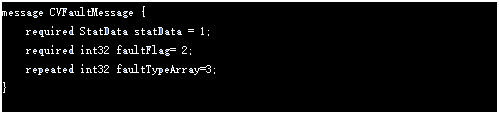


statData：统计数据包

faultFlag：故障标识

faultTypeArray：故障数组

1. CVFaultMessage数据

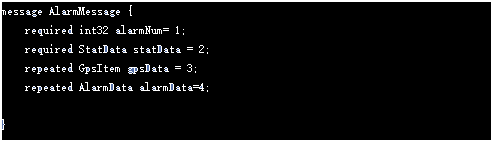


statData：统计数据包

faultFlag：故障标识

faultTypeArray：故障数组

1. AlarmMessage数据



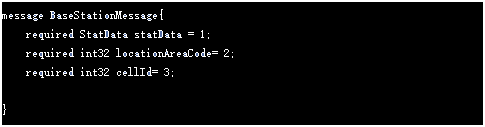
alarmNum：警情编号

statData：统计数据包

gpsData：GPS数据

alarmData：警情消息

1. BaseStationMessage数据



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



utcTime ：设备当前UTC时间

gpsItem ：1个包的GpsItem 结构

参见《OBD 数据结构定义（Protobuf）》