浏览器到Node端接口约束文档

1、 背景说明

软件二部在浏览器到后端server中间做了node中间层，做数据转发、文件传输中转、接口数据处理等操作。针对浏览器到Node端做对应的接口约束，规范开发流程和逻辑。

2、 规范说明

2.1、URI规范

1、server端模块划分

(1)、对于页面特有的接口，以页面为维度划分; (2)、对于公共的模块，比如车辆、报警类型以实体为维度来划分；

2、node端命名规范

(1)、server端的api和service的js文件统一采用驼峰命名 (2)、相同页面或者相同模块的api与service命名尽可能保持一致，方便定位 (3)、定义node端接口的path，以'/api/对应的service文件名/....'命名，方便前端调用接口明确知道调用的哪个service或者api文件的接口，示例如下：

```
// 页面业务功能：    /api/pageName/queryServiceForPage
// 公共功能：       /api/common/upload
```

2.2、版本规范(可选)

统一将版本号放入URI中,统一在域后面第二节填充版本。

Eg: http://ip:port/api/v1/vehicle

2.3、请求方式规范

目前HTTP协议中有8种方法，我们系统中目前常用的主要有4种，目前着重先说明这四种，至于其他的后续扩展有需要再进行补充说明

请求方式	场景	说明
GET	查询	从服务器获取资源（一项/多项）
POST	添加/查询/终端交互	在服务器创建一个资源，或者查询，在参数过多的情况下可以使用
PUT	更新	在服务器更新资源，客户端提供更新属性等信息
DELETE	删除	从服务器删除资源信息

GET请求：

	示例	备注
过滤条件	?type=1&age=16	允许一定的uri冗余，如 /zoos/1 与 /zoos?id=1
排序	?sort=age&order=asc	指定返回结果按照哪个属性排序，以及排序顺序
投影	?whitelist=id,name,emai	
分页	?pageNum=2&pageSize=100	指定第几页，以及每页的记录数

如果请求参数过长，get请求满足不了的情况下建议直接用json封装，通过body直接post提交

POST请求：

提交方式为POST时，交互参数统一封装为json的方式进行提交

Content-type: application/json

2.4、内容命名规范

1、参数名称和返回字段的命名统一采用驼峰，如：userId, companyName, pageNum 等。

2、Header 大写开头，分词采用'-'和http标准请求头保持一致，如 **Accept-Encoding**，**Connection**等

2.5、权限验证规范

为了方便在整个应用交互中权限控制和访问域控制，需要在请求中携带认证信息，请求头标示统一为Token。Token的构成形式为 userId, roleId, clientId, utcTimeStamp

总共分为4部分

userID	**系统用户id，由web统一管理分配，是系统用户唯一标示符**
roleId	角色ID，用户所属角色的ID，是角色唯一标示符，通过角色ID能够界定用户操作权限范围
clientId	业务模块编码，用于标示当前请求是来源于哪个模块进程，方便服务端区分消费业务进程，如DST
utcTimeStamp	token生成时的时间戳，方便控制token有效期

将4部分通过半角逗号分隔，组装成字符串用过des加密。在用户登录阶段由服务端返回给客户端供后续业务接口使用。

在业务查询中，如果调用方没有指定查询范围时，默认采用访问域内的所有范围按照参数规则检索并返回应答信息

2.6、应答规范

目前应答统一为json数据格式，内容命名规范同2.4

应答结果封装

为了方便整体在上下级网元间的正确信息传导，针对应答数据进行统一封装。封装结构主要由两部分构成。

code: 用于描述服务器处理情况，是否正常处理（参数格式是否正确、权限信息是否满足、服务器是否出现异常等情况），

msg: 内部信息，主要用于内部信息提示，不用于用户界面信息展现，比如内部验重失败的原因，参数错误的原因，服务器异常的原因等，方便使用方明确问题

data: 部分则为数据应答部分内容。所有结果在第一层级只有这两个信息。

```
// Eg:
// 无数据
{
  "code":200,
  "msg": "xxxxxxxx",
  "data":[] // 一定要求后端为空情况返回空数组
}
// 正常返回数据
{
  "success": true,
  "code":200,
  "msg":"xxxxxxxx",
  "data":{
    "dataCount":500,
    "dataList":[{…},{…}]
  }
}
```

2.6.1、常用true/false应答

此应答通常用于保存、删除、修改、指令下发等场景中，返回应答只需要告知调用方执行结果即可，应答格式为

```
{
  "code": 500,
  "success": false,
  "msg": "xxxxxxxx",
  "data":[] // 一定要求后端为空情况返回空数组
}
```

2.6.2、分页应答

此应答主要用于分页查询的场景下用于数据返回，其中主要包括了当前查询数据的总条数、当前分页获取的信息

```
{
  "code":200,
  "msg":"xxxxxxxx",
  "data": {
    "total": 500,  // 数据总数
    "pageSize": 10, // 分页条数
    "pageNum": 1, // 当页是第一页
    "dataList": [{…},{…}]
  }
}
```

2.6.2、创建应答

此应答作为创建数据时的通用应答，主要包括了当前创建信息的唯一标识和创建结果

```
{
  "success": true,
  "code":200,
  "msg":"xxxxxxxx",
  "result":{
    "id": 1,
    "saveResult": true
  }
}
```

id: 创建数据的唯一标识

saveResult: 保存结果，true表示保存成功

2.7、常见错误码定义

模块	错误码	描述
通用	200	服务器正常处理，没有出现异常、参数错误等情况
201	权限校验失败，当前权限不能操作响应请求	
202	服务器异常，服务器处理过程中出现异常，整个流程未能正常进行	
203	请求参数异常	
204	唯一校验不通过	
205-400	预留	
终端状态	401	终端不在线
402	终端响应超时	
403	终端应答异常	
404-500	预留	
用户	1000-1099	该模块自定义预留
角色	1100-1199	该模块自定义预留
车组	1200-1299	该模块自定义预留
车辆	1300-1399	该模块自定义预留
终端	1400-1499	该模块自定义预留
sim卡	1500-1599	该模块自定义预留
驾驶员	1600-1699	该模块自定义预留
报警	1700-1799	该模块自定义预留
证据	1800-1899	该模块自定义预留