# API规范

## 一、rest风格

###### Rest：Representation Status Transfer (表述性状态转移)

Rest风格是针对于资源的定位方式而设计的，然而客户端、服务端之间的通讯核心其实也是资源的调度过程；那么rest恰恰满足这种需求。Rest api的设计规范如下（示例：user — order）：

|  |  |  |  |
| --- | --- | --- | --- |
| 资源 | 请求方式 | 响应 | 描述 |
| /v1/users | POST | 201 | 用户信息的添加 |
| /v1/users/{id} | GET | 200 | 查询指定用户信息 |
| /v1/users/{ id } | PUT | 200 | 更新指定用户信息 |
| /v1/users/{ id } | DELETE | 200 | 删除指定用户信息 |
| /v1/users/{ id }/orders/{ id } | GET | 200 | 获取指定用户的订单ID为1的订单信息 |

其实rest规范远不止这些，还有：***Http缓存、Cache、Rate-Limit、Hateoas、安全性（Header、Body）、Error、一致性(Accept/Content-Type、Action、ETag、Status)***

下面对比关于rest api与目前的非rest api进行简单对比如下（示例：user — order）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 需求 | 非Rest api | Rest api | 非Rest优点/缺点 | Rest优点/缺点 |
| 1.查询用户的所有订单;  2.查询该订单所属用户 | 1./user/findOrderByUserId  2./order/findUserByOrderId | 1./users/{id}/orders  2./orders/{id}/users | 优点：  抒写方便、随意性强  缺点：  不易于扩展、可读性差、后期维护不好 | 优点：  资源层次清晰、扩展性强、url映射强  缺点： |
| 针对于所有更新操作的拦截进行日志收集 | 非rest风格api的拦截进行日志落盘必须的涉及到业务代码进行拦截（因为只有POST/GET方式） | Rest风格api拦截进行日志落盘只需要进行请求方式的拦截就可以（因为提供POST/GET/PUT/PATCH/DELETE方式） | 优点：  缺点：  代码入侵性强、不易扩展 | 优点：  代码入侵性弱、易于扩展、分离  缺点： |

上面是简单介绍了rest规范，那么对于客户端来说需要作出的调整是：

|  |  |  |
| --- | --- | --- |
| 调整项 | 非Rest api请求方式 | Rest api请求方式 |
| url地址 | url地址?param1=value1& param1=value1 | 使用资源定位进行url的拼接:/v1/users/{id}?param1=value1& param1=value1 |
| 请求method | POST/GET | POST/GET/PUT/PATCH/DELETE |

## 二、传输：https、加密、压缩

###### 采用https传输协议

**说明：**

HTTP协议以明文方式发送内容，不提供任何方式的数据加密，如果攻击者截取了Web浏览器和网站服务器之间的传输报文，就可以直接读懂其中的信息。HTTPS在HTTP的基础上加入了SSL协议，SSL依靠证书来验证服务器的身份，并为浏览器和服务器之间的通信加密。

**客户端改动：**

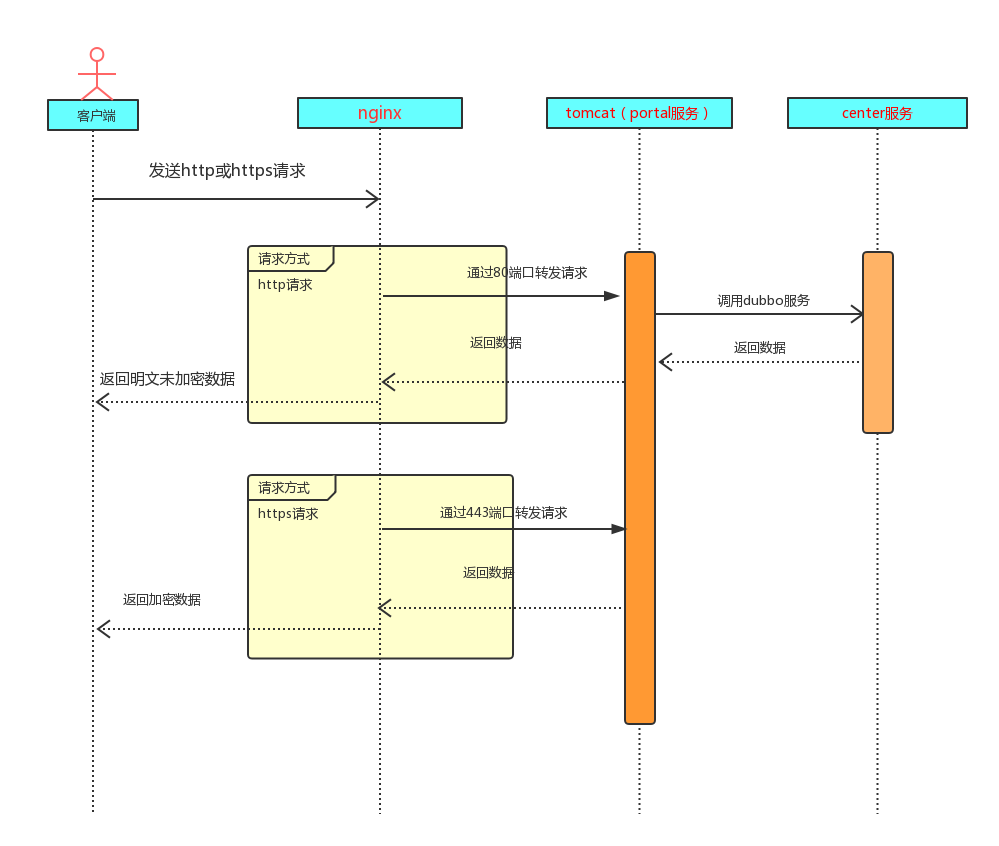
需要客户端将之前的http域名换成https即可，其它不用动。兼容老版本的接口数据返回。

**与之前对比：**

1、之前的http是明文传输，很容易被截取到参数信息，https数据更安全，第三方截取不到请求参数，无法模拟请求到我们的服务器，保证服务安全。

2、谷歌对使用https的网址或app有一定的排名提升。

**流程图如下：**



###### 接口数据返回加密

**方案：**

    服务端会下发两个key：accessKey和secretKey给客户端，一个公钥一个秘钥。其中 accessKey相当于用户标识，会通过它区分不同调用方，而secreKey相当于提供给用户的密码，在接口使用过程中都不会在网络中传输，只有用户和服务端知道，客户端用于解密数据，所有接口一个秘钥 。

    利用请求头信息的token字段,请求到服务端，会进行token校验，校验通过才会返回数据。

    token的定义规则如下

1. String token = "时间戳" +":" +accessKey ;

    对返回的数据进行AES加密，避免第三方获取到数据的内容，服务端和客户端公用一个秘钥，加密数据并进行Gzip压缩返回，这个要看接口数据的安全程度，数据量小的不用压缩。

**客户端改动：**需要先解压缩数据，再解密数据，才能拿到真正的数据，真正的数据和之前返回保持一致。  
**与之前的方式对比：**

1、https保证了数据传输的安全，数据加密保证了应用层的数据安全，通过两种方式保证了整体数据的安全性。

2、加密和压缩的缺点是增加了接口的操作耗时，所以要视接口数据的重要程度和数据量大小，确定是否进行加密或压缩操作，整体上在此方案内。

3、token新增时间戳和公钥，服务端可以利用token的校验规则，来防止一定的接口攻击，保证接口不会被第三方恶意调用。

**数据流程 如下：**

