[1 文档介绍](#_Toc28685)

[1.1 文档目的](#_Toc11744)

[1.2 适用范围](#_Toc16390)

[1.3 术语与解释](#_Toc15909)

[1.4 文档约定](#_Toc8130)

[2 背景概述](#_Toc30073)

[3 需求分析](#_Toc3854)

[4 系统设计](#_Toc24196)

[4.1 系统总体架构](#_Toc22895)

[4.1 认证](#_Toc13049)

[4.2.1 Jwt简介](#_Toc29926)

[4.2.2 认证流程](#_Toc31538)

[4.2 鉴权](#_Toc7433)

[4.3.1 鉴权流程](#_Toc3712)

[4.3 认证鉴权时序图](#_Toc5524)

[5 接口设计](#_Toc3233)

[5.1 登录接口](#_Toc26256)

[5.1.1 接口说明](#_Toc12205)

[5.1.2 接口url](#_Toc21256)

[5.1.3 接口入参](#_Toc8849)

[5.1.4 接口请求示例](#_Toc9749)

[5.1.5 接口出参](#_Toc10332)

[5.1.6 响应示例](#_Toc4479)

[5.2 鉴权接口](#_Toc736)

[5.2.1 接口说明](#_Toc29263)

[5.2.2 接口url](#_Toc15226)

[5.2.3 接口入参](#_Toc13058)

[5.2.4 使用示例](#_Toc18174)

[5.2.5 接口出参](#_Toc62)

[5.2.6 返回示例](#_Toc1560)

[5.3 token重刷接口](#_Toc27696)

[5.3.1 接口说明](#_Toc28065)

[5.3.2 接口url](#_Toc15885)

[5.3.3 接口入参](#_Toc16061)

[5.3.4 使用示例](#_Toc22078)

[5.3.5 接口出参](#_Toc13498)

[5.3.6 返回示例](#_Toc5437)

[5.4 登出接口](#_Toc32513)

[5.4.1 接口说明](#_Toc25148)

[5.4.2 接口url](#_Toc20391)

[5.4.3 接口入参](#_Toc6144)

[5.4.4 使用示例](#_Toc19287)

[5.4.5 接口出参](#_Toc4506)

[5.4.6 返回示例](#_Toc19882)

[5.5 错误码](#_Toc7651)

# 文档介绍

## 文档目的

本文档是对tsp认证和鉴权做的顶层设计，主要包含对认证模块在整个系统中的层次，以及网关在整个认证鉴权体系中承载的角色进行的设计。本阶段，文档主要用于指导网关和用户认证鉴权相关项目的开发。

## 适用范围

研发部设计、开发、测试和所有被授权人员

## 术语与解释

|  |  |
| --- | --- |
| **缩写、术语** | **解 释** |
| TSP | Telematics Service Provider，特指车联网平台 |
| ms-tsp-user-auth | 用户认证鉴权相关服务 |
| gateway | 网关，默认2C |

## 文档约定

* 接口通用约定

后台提供GET和POST两种类型的请求;基于restful风格接口设计规范。后端返回采用统一的json数据结构，必定包含respCode,respMsg,body三个字段.结构示例如下：

{

“respCode”:”00000”,

“respMsg”:”Success”

“respData”:null

}

* 错误码约定

错误码为字符串类型，共5位，分成两个部分：错误产生来源+四位数字编号。 说明：错误产生来源分为A/B/C，A表示错误来源于用户，比如参数错误，用户安装版本过低，用户注册错误等问题；B表示错误来源于当前系统，往往是业务逻辑出错，或程序健壮性等问题；C表示错误来源于第三方服务，比如极光推送服务出错，消息投递超时等问题；四位数字编号从0001到9999，大类之间的步长间距预留100。

# 背景概述

TSP平台主要提供车联网运营业务功能，对外提供多端业务访问：车主手机APP或Pad、车主网站、运营管理网站、大屏展示以及对车辆HU的业务支持。TSP平台在业务上是一个具有海量用户和数据的综合性服务平台。既有百万级用户基于HTTPS的低延迟请求，也有百万级网联终端基于TCP长连接的实时数据处理，更有海量异构数据的存储和计算，所以平台的高性能和高扩展性的设计变得尤为重要，这就需要在架构上从顶层规划，分而治之，基于业务模块对系统进行合理化拆分，做到高内聚、低耦合，隔离业务变化点和预计可能的性能瓶颈点。

因此需要针对tsp需求设计一套高可用，高性能的网关和针对用户进行认证和鉴权的服务。

# 需求分析

HTTP 是一个无状态的协议，一次请求结束后，下次在发送服务器就不知道这个请求是谁发来的了（同一个 IP 不代表同一个用户），在 Web 应用中，用户的认证和鉴权是非常重要的一环。为了支撑tsp平台对流量，系统安全和技术可扩展的要求，现需要设计一套认证和鉴权的微服务。

# 系统设计

## 4.1 系统总体架构

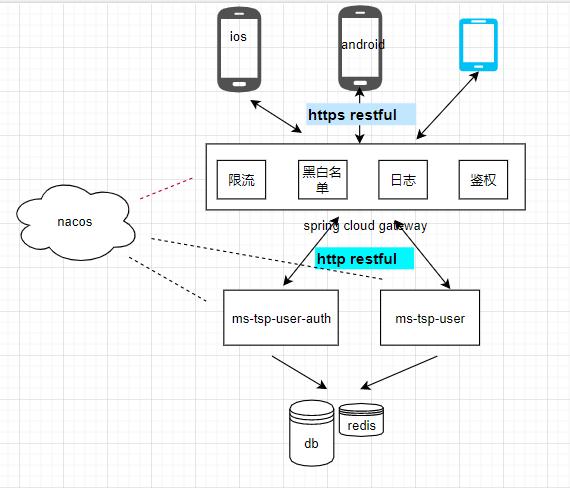


图1 系统架构图

* Spring cloud gateway 网关

gateway网关承载tsp平台路由，限流，日志，黑白名单控制的作用;同时网关需要对鉴权进行拦截，调用ms-tsp-user-auth进行鉴权成功后，修改http请求头，写入必要信息后将请求路由到对应的服务提供者。

* Nacos

提供tsp平台中服务注册中心和配置中心的功能。

* ms-tsp-user-auth

提供登录，登出，鉴权等服务

* ms-tsp-user

提供用户相关服务，包括设置，获取，修改个人信息;忘记密码，修改密码等用户相关服务。

## 认证

认证主要完成用户身份识别和用户可访问资源的提取和缓存。当用户填写用户名，密码等相关认证信息后，请求后端进行认证处理。该请求经gateway直接路由到ms-tsp-user-auth服务，由ms-tsp-user-auth服务对参数进行校验，校验成功后生成accessToken和refreshToken并返回.token主要采用jwt相关协议进行生成。

### 4.2.1 Jwt简介

JSON Web Token（JWT）是一个开放的行业标准（RFC 7519），它定义了一种简介的、自包含的协议格式，用于在通信双方传递json对象，传递的信息经过数字签名可以被验证和信任。JWT可以使用HMAC算法或使用RSA的公钥/私钥对来签名，防止被篡改。

传统授权方法的问题是用户每次请求资源服务，资源服务都需要携带令牌访问认证服务去校验令牌的合法性，并根据令牌获取用户的相关信息，性能低下。

使用JWT的思路是，用户认证通过会得到一个JWT令牌，JWT令牌中已经包括了用户相关的信息，客户端只需要携带JWT访问资源服务，资源服务根据事先约定的算法自行完成令牌校验，无需每次都请求认证服务完成授权。

* Jwt token报文如下：

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1NDQ1MTE3NDMsImp0aSI6IjYxYmVmNjkyLTE4M2ItNGYxYy1hZjE1LWUwMDM0MTczNzkxOSJ9.CZzB2-JI1oPRFxNMaoFz9-9cKGTYVXkOC2INMoEYNNA

* Jwt结构图如下：

signature

payload

header

Jwt包含头部（Header）、负载（Payload）、签名（Signature）三部分，头部和负载以 JSON 形式存在，这就是 JWT 中的 JSON，三部分的内容都分别单独经过了 Base64 编码，以 . 拼接成一个 JWT Token。

* **header**

JWT 的 Header 中存储了所使用的加密算法和 Token 类型。

{

“alg”:”hs256”,

“typ”:”jwt”

}

* **Payload**

Payload 是负载，JWT 规范规定了一些字段，并推荐使用，开发者也可以自己指定字段和内容。

jti：jwt的唯一ID，使用uuid算法生成，必须

rti：refresh-token的jti，作为登录成功的sessionid，也是缓存数据key的重要部分

exp：失效时间，使用unix时间戳，如 "exp":1300819380。实际使用时，可以在此时间后宽限5分钟。refresh-token的有效期为30天。access-token的有效期为4小时。必须

uid：用户ID，业务属性

aid：账户ID，业务属性

lid：登录账户，业务属性

lty：登录账户类型，业务属性

did：设备ID，业务属性。

dty：设备类型，业务属性。

业务属性暂定，根据业务需求调整，作为登录session的重要数据项缓存

* **signature**  
  Signature 部分用来验证 JWT Token 是否被篡改，所以这部分会使用一个 Secret 将前两部分加密，逻辑如下。

HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)

### 4.2.2 认证流程

认证流程主要是在ms-user-auth服务中完成的，网关接收到该请求后只处理通用的拦截逻辑，不做特殊的处理。直接将原始请求路由到ms-user-auth服务中，由该服务提供的认证接口进行处理。具体流程如下图：

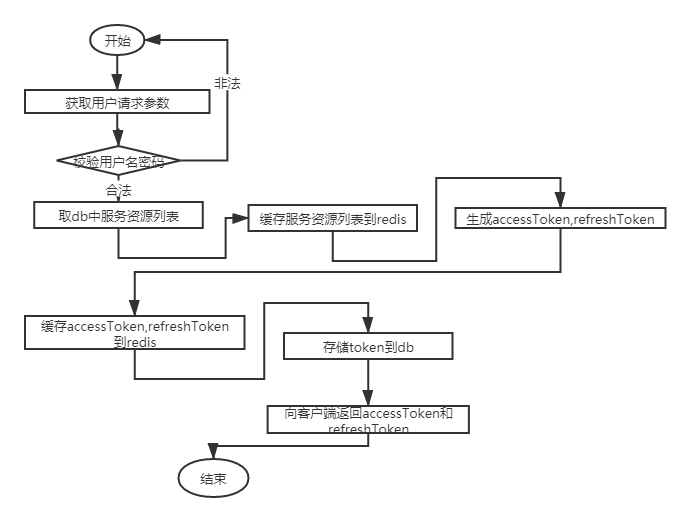


图2 认证流程图

流程处理说明

1. ms-tsp-user-auth服务接收到请求后，获取请求参数，查询db中用户数据
2. 对用户的密码进行sha256加盐进行加密，并和数据库数据进行校验
3. 校验非法直接返回，否则执行4
4. 从db中获取用户服务清单
5. 将服务清单缓存到redis
6. 根据jwt算法生成accessToken,refreshToken;accessToken设计过期时间为4hour;refreshToken设计过期时间为7day
7. 缓存accessToken,refreshToken到redis
8. 存储accessToken,refreshToken到db
9. 向客户端返回accessToken和refreshToken
10. 客户端登录成功后需要将后端的accessToken和refreshToken进行缓存

ms-tsp-user-auth中redis中间件作用如下：

1)缓存用户服务列表

2)缓存accessToken和refreshToken

## 鉴权

鉴权主要是完成验证用户是否认证，已经用户对应资源的访问权限控制。当客户端访问后端资源时，请求到达网关，由网关进行拦截，网关获取客户端访问资源，组装鉴权相关信息，调用ms-tsp-user-auth提供服务接口。当鉴权成功校验后向网关返回用户信息。网关需要对鉴权成功后的响应做修改http herader的处理，添加必要信息后将原始请求路由到对应的服务。

### 4.3.1 鉴权流程

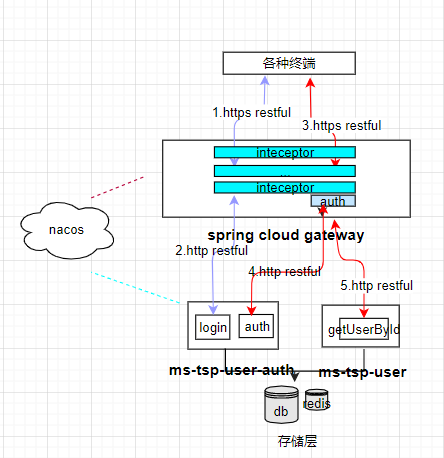


图3 鉴权流程图

鉴权流程说明

1. 客户端发起请求用户信息查询请求，调用后端服务
2. 网关auth拦截请求后获取所要访问服务资源路径，调用ms-user-auth鉴权接口进行鉴权
3. ms-tsp-user-auth接收到请求后校验accessToken的合法性，从redis从获取accessToken和服务列表并进行accessToken和服务资源的检查，校验通过后返回用户相关信息
4. 网关接收到响应后判断是否鉴权成功，失败的话直接返回;如果成功则添加必要的header头信息后将请求转发到ms-tsp-user服务，调用查询用户信息接口并返回数据

## 认证鉴权时序图

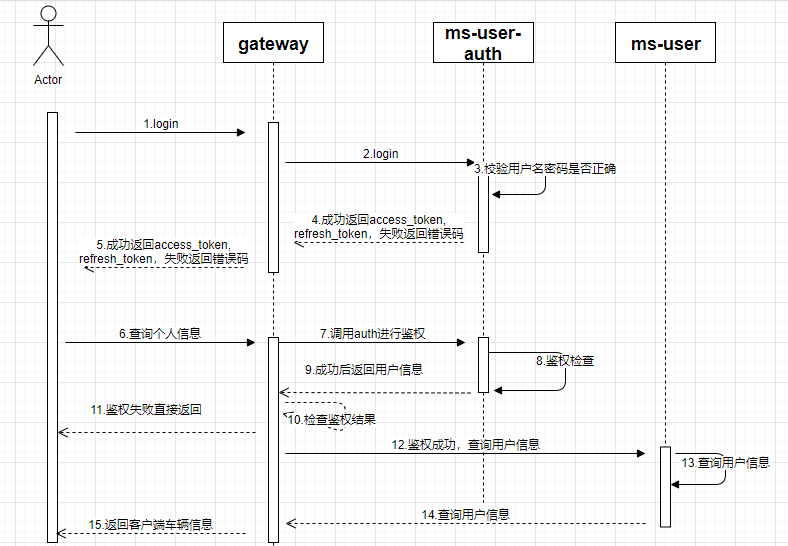
****

图4 认证鉴权时序图

详细流程如下：

1. 客户端发送基于userName,password向网关发送login请求，网关根据配置不进行拦截;
2. 网关将登录请求转发到用户登录服务接口
3. 用户服务获取用户名，密码，校验密码是否匹配;如果匹配成功，则根据jwt生成算法，生成对应accessToken,refreshToken并返回;否则向调用端返回错误码。
4. 返回jwt或者错误码到网关
5. 网关向客户端返回调用结果
6. 客户端在http请求头Authorization添加登录返回的accessToken向gateway发送查询用户信息
7. 网关接收到请求检查header中Authorization是否包含accessToken,使用jwt算法进行token合法性和时效性校验，如果校验非法直接返回，否则调用ms-tsp-user-auth进行鉴权
8. ms-tsp-user-auth接收到请求后，进行token合法性检查，检查通过后调用redis检查token是否存在和服务权限校验，校验通过返回用户相关信息
9. ms-tsp-user-auth返回鉴权结果
10. 网关接收到鉴权响应结果后检查鉴权结果,进行相应的处理。如果成功执行12操作，如果失败执行操作11
11. 鉴权失败直接返回
12. 鉴权成功，调用ms-tsp-user-auth查询用户相关细细
13. ms-tsp-user-auth接收到请求后处理请求
14. 返回用户信息
15. 网关返回客户端调用

# 接口设计

## 登录接口

### 5.1.1 接口说明

提供登录功能，检验登录相关参数，检验成功生产jwt并返回;否则返回相关错误码告知终端进行相关后续操作。

具体操作步骤如下：

1）客户端需要将用户输入密码进行md5后传给后端，后端根据取到的密码进行sha256加盐进行加密和服务端存储的密码进行校验

2）校验失败，直接返回错误码否则执行步骤3

3）从db中获取用户服务清单

4）将服务清单缓存到redis

5）根据jwt算法生成accessToken,refreshToken;accessToken设计过期时间为2hour;refreshToken设计过期时间为7day

6）缓存accessToken,refreshToken到redis

7）存储accessToken,refreshToken到db

8）向客户端返回accessToken和refreshToken

9）客户端登录成功后需要将后端的accessToken和refreshToken进行缓存

### 5.1.2 接口url

POST /tspUserAuth/v1/login

### 5.1.3 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| account | string | Y | 用户名 |
| password | string | Y | 密码，32位md5值，前端需要进行md5后传给后端 |
| appType | int | Y | 应用类型 |
| deviceType | int | Y | 设备类型  1-andorid  2-ios |
| afs | string | N | 风控参数，人机校验，防止机器刷 |
| fingerprint | string | N | 设备指纹 |

### 5.1.4 接口请求示例

{

“account”:”18027888111”,

“password”:”xxx”,

“appType”:1,

“deviceType”:1

}

### 5.1.5 接口出参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| accessToken | string | Y | 鉴权token,客户端登录拿到此token后需要缓存，每次调用后端接口需要在header中增加此token |
| refreshToken | String | Y | 当后端返回302错误码时，用此数据调用后端提供的tokenRefresh接口进行token重刷 |

### 5.1.6 响应示例

{

“respCode”:200,

“respMsg”:”success”

“respData”:{

“accessToken”:”jwtToken”,

“refreshToken”:”jwtToken”}

}

## 鉴权接口

### 5.2.1 接口说明

鉴权接口提供给网关调用。当客户端请求访问后端的资源接口时，网关拦截请求，获取客户端将要访问的资源，然后调用此接口组装接口相关的参数进行鉴权操作。

具体鉴权步骤如下：

1. 客户端将accessToken加入到header的Authorization字段中，访问后端服务资源
2. 网关拦截请求，获取相应的服务资源，调用此接口，传递必要的参数进行鉴权操作
3. 此接口接收到请求后，校验token的合法性
4. 当校验accessToken过期后返回A0303错误码，否则执行步骤6
5. 客户端获取到该错误码后需要调用/auth/tokenRefresh接口进行accessToken重刷，拿到accessToken后需要执行缓存操作,并重新执行1
6. 鉴权服务接口接收到请求后取redis中缓存token和服务资源清单进行token和权限的校验
7. 校验成功后返回返回用户相关信息，否则执行8
8. 网关根据鉴权返回结果进行处理。成功则重写header头相关信息，并路由原始请求
9. 返回客户端相关错误码

### 5.2.2 接口url

GET /tspUserAuth/v1/check

### 5.2.3 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| servicePath | string | Y | 服务资源路径 |

### 5.2.4 使用示例

http://localhost:8080 /tspUserAuth/v1/check?servicePath=xxx

### 5.2.5 接口出参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| userId | long | Y | 用户id |

### 5.2.6 返回示例

{

“respCode”:200,

“respMsg”:”success”

“respData”:{

“userId”:1000001

}

}

## token重刷接口

### 5.3.1 接口说明

重新获取accessToken接口。当客户端调用后端接口返回A0302时，客户端检查本地缓存refreshToken是否过期，如果没有过期，需要重新调用此接口并将返回结果进行缓存。此后请求服务资源时需要将本地缓存的accessToken加入到header的Authorization字段中，访问服务资源.

具体实现流程：

1. 客户端接收到响应码A0302时，需要取本地缓存的refreshToken调用此接口进行token重刷
2. 服务端接收到请求后校验refreshToken的合法性
3. 查询redis缓存检验refreshToken
4. 检验通过后通过jwt算法重新生成accessToken
5. 缓存accessToken到redis
6. 成功返回accessToken，客户端收到响应后需要重新将token写入到本地缓存
7. 返回A0301,客户端接收到请求后需要跳转到登录页

### 5.3.2 接口url

GET /tspUserAuth/v1/tokenRefresh

### 5.3.3 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| refreshToken | string | Y | 登录时返回的刷新token |

### 5.3.4 使用示例

http://localhost:8080 /tspUserAuth/v1/tokenRefresh?refreshToken=xxx

### 5.3.5 接口出参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| accessToken | string | Y | 鉴权使用token,客户端需要将此数据缓存，每次请求后端接口需要在header中Authorization总增加此参数 |

### 5.3.6 返回示例

{

“respCode”:200,

“respMsg”:”success”

“respData”:{

“accessToken”:”jwtToken”

}

}

## 登出接口

### 5.4.1 接口说明

登录态下调用此接口进行退出操作，此接口调用后用户访问后端需要鉴权接口时需要重新登录。

1. 校验accessToken的合法性
2. 查询redis相关缓存，获取redis中token,服务清单，清理相关缓存资源
3. 成功返回，失败返回相应错误码
4. 客户端接收到响应后需要做相应的处理，成功需要清理本例token缓存。

### 5.4.2 接口url

GET /tspUserAuth/v1/logout

### 5.4.3 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |

### 5.4.4 使用示例

http://localhost:8080 /tspUserAuth/v1/logout

### 5.4.5 接口出参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |

### 5.4.6 返回示例

{

“respCode”:”00000”,

“respMsg”:”success”

“respData”:null

}

### 5.5 三方登录接口

#### 5.5.1 接口说明

三方登录接口,提供内部调用.

1. 客户端获取三方授权码code
2. 客户端调用/v1/third/auth授权服务端进行资源获取
3. 服务端从三方获取资源访问token
4. 服务端通过token获取用户基本资源信息和openId
5. 服务端检测用户三方openId是否绑定，如果已经绑定则进行6；否则进行7相关操作
6. 调用本接口进行三方登录；服务端接收到请求后校验thirdToken合法性，如果合法完成认证过程；否则返回相应错误码
7. 执行绑定操作；绑定成功后执行6

#### 5.5.2 接口url

POST /v1/third/login

#### 5.5.3 接口入参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| thirdToken | string | Y | 三方auth时返回的token |

#### 5.5.4 请求示例

{

“thirdToken”:”18310411954”

}

#### 5.5.5 接口出参

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| accessToken | string | Y | 资源访问令牌 |
| refreshToken | string | Y | 认证通过发放的令牌，用于刷新accessToken |

#### 5.5.6 返回示例

{

“respCode”:”00000”,

“respMsg”:”Success”,

“respData”:{

“accessToken”:””,

“refreshToken”:””

}

“respTime”:” 2020-06-12 10:53:13”

}

## 错误码

|  |  |
| --- | --- |
| 错误码 | 说明 |
| A0201 | 用户账号不存在 |
| A0202 | 用户账户被冻结 |
| A0210 | 用户密码错误 |
| A0211 | 用户输入密码次数超限 |
| A0301 | 访问未授权 |
| A0302 | token过期需要进行refresh |
| A0501 | 请求次数超过限制 |