

2-4: OOMALL 服务模块设计说明书

1. 前言	2
1.1 目的	2
1.2 预期读者	2
1.3 项目背景	2
2. 系统概述	2
2.1 系统目标	2
2.2 运行环境	2
2.2.1 硬件平台	2
2.2.2 软件环境	2
2.3 系统总体结构	2
2.4 包	3
3. 对象模型	4
3.1 领域对象	4
3.2 对象状态	5
3.2.1 服务商 ServiceProvider 状态	5
3.2.2 服务单 ServiceOrder	5
3.2.3 商户服务 ShopService	6
4. 数据库	7
5. 程序逻辑	8
5.1 服务商	9
5.1.1 服务商修改服务商状态	9
5.2 服务	12
5.2.1 服务商定义在某个地区为某种商户的服务	12
5.3 服务单	14
5.3.1 创建服务单	14

1. 前言

1.1 目的

描述 OOMALL 系统服务模块的设计思路以及代码的结构，方便协调开发工作和后期维护。

1.2 预期读者

相关模块的开发人员。

1.3 项目背景

本项目是厦门大学信息学院软件工程专业《软件工程》《面向对象分析与设计》和《JavaEE 平台技术》三门课程的联合课程设计。支付模块的设计和实现开始于 2022 年。2022 年秋季学期由软件工程专业 2020 级学生完成第一次迭代，2023 年秋季学期由软件工程专业 2021 级学生开始第二次迭代。

2. 系统概述

2.1 系统目标

实现系统中服务模块的相关功能。

2.2 运行环境

2.2.1 硬件平台

华为云耀云服务器集群。

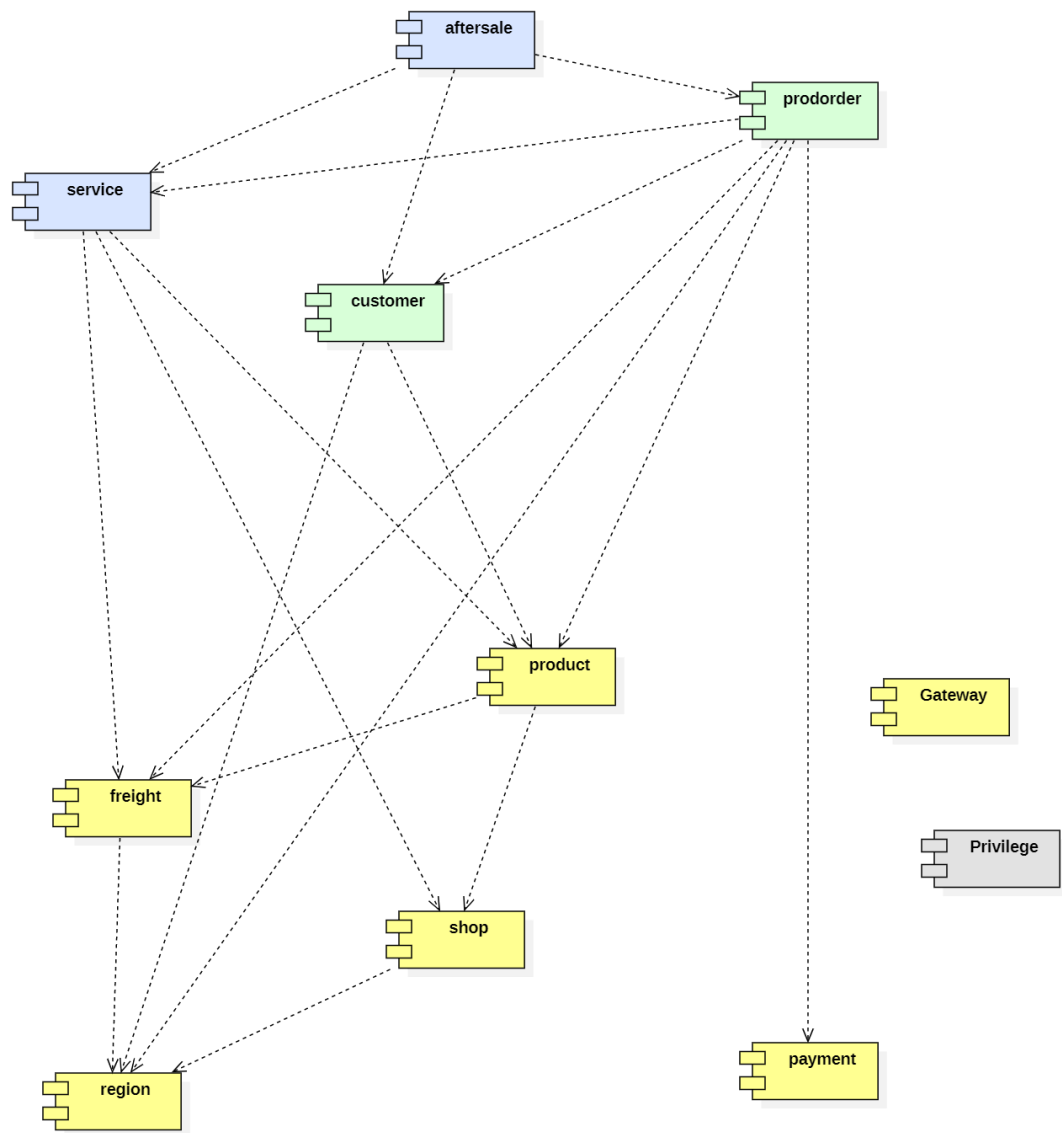
2.2.2 软件环境

操作系统：Ubuntu 22.04

支持环境：Docker Swarm, Tomcat 10.0.23

数据库：MySQL, Mongo

2.3 系统总体结构



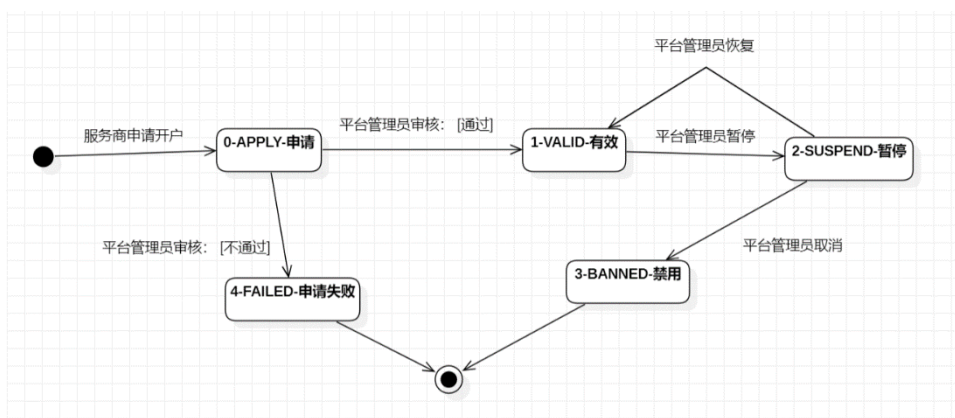
2.4 包

说明：

- ① 同时出现 orderitem 和 productitem 是将前者调后者的职责 copy 过来
- ② productitem 是提了售后或者购买商品后就产生了服务单
- ③ product 和 service 画在对象模型是因为要描述：服务商建立和服务的组合关系然后再由商铺指定商品和服务的关系

3.2 对象状态

3.2.1 服务商 ServiceProvider 状态



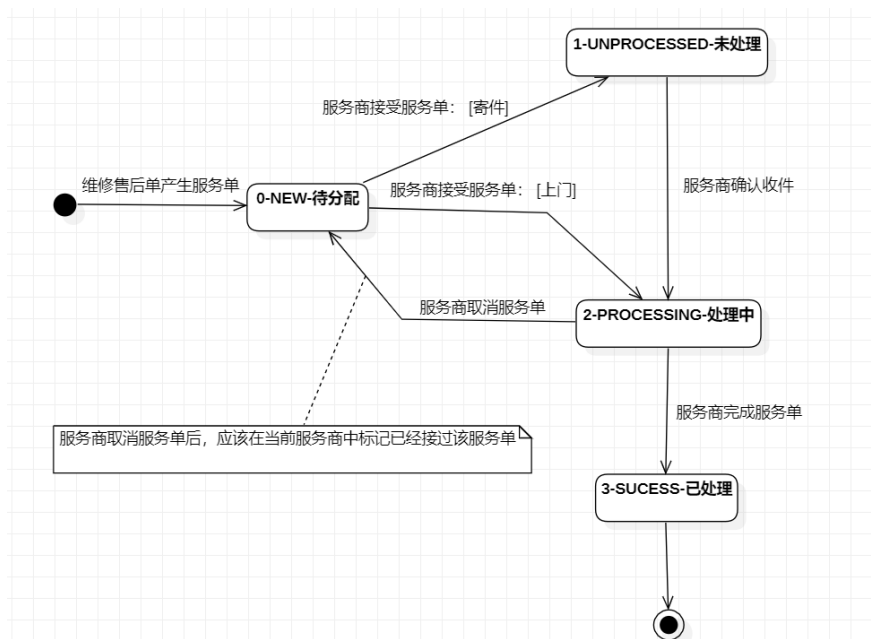
说明：

服务商申请开户注册服务商账户时，该服务商处在 APPLY 申请态，需要经过平台管理员审核：

若审核服务商通过，则服务商变为 VALID 有效态

若审核不通过后给出失败原因，直接结束，服务商需要再次提交开户申请（说明：用例无修改服务商的申请开户信息，因此此时实现是需要服务商需要重新申请开户）

3.2.2 服务单 ServiceOrder



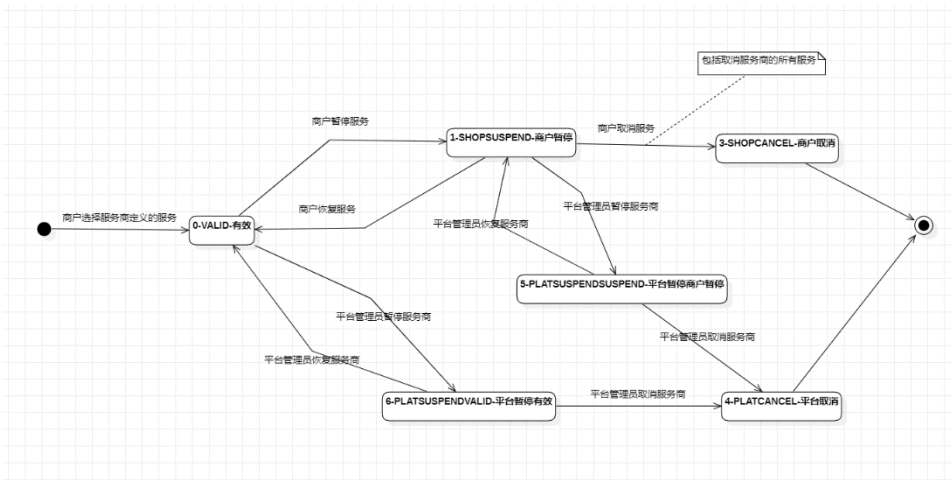
说明：

系统目前只实现售后服务，即从一个维修（寄件维修或者上门维修）类型的售后单产生服务单，当服务单创建出来后是一个 0-NEW-待分配的状态。当具有接单资质的服务商选择接单后，根据服务单类型（寄件服务还是上门服务）分别迁移到 3-UNPROCESSED-未处理和 2-PROCESSING-处理中。此处区分服务单类型的主要原因是寄件维修的服务需要经过物流运输到服务商确认收件的过程，需要通过从未处理到处理中标识维修物品已经到服务商。

当服务商完成服务单则将 PROCESSING 处理中的服务单标记为 SUCCESS 已处理；服务商只能取消处理中的服务单：一个是上门无法与顾客达成共识将处理中直接取消；其次是服务商确认收件后联系顾客无法达成共识也将处理中直接取消，这也是区分寄件和上门的原因

当服务商取消服务单后，该服务单会重新变成 NEW 待分配状态，给其他的服务商接单

3.2.3 商户服务 ShopService



说明：

商户只能选择服务商定义的**有效服务 Service**，处于无效的服务不能被挑选。一旦选择服务后该商户服务即处于 VALID 有效态：

(1) 只考虑商户操作下的状态迁移不考虑平台管理员的操作：

商户对于处在 VALID 有效态的服务可以选择暂停后变为 SHOPSUSPEND 商户暂停，

商户只能对处在 SHOPSUSPEND 商户暂停状态的下的服务进行取消

商户不能直接取消有效态的商户服务，要先暂停保证先前已接的服务单都被处理完或者取消等才能取消，否则将有效态的商户服务直接取消将导致某些已被接单的服务单无法继续关联商户

此时只有取消该商户服务的商户不能再继续使用该服务商定义的服务；其他商户若选择的也是该服务商定义的服务，则还是可以使用。

(2) 考虑只有在平台管理员对 VALID 的商户服务的操作

平台管理员暂停服务商后，所有当前选择该服务商定义的服务都将处在 PLATSUSPENDVALID 平台暂停有效态，当此时平台恢复该服务商时，原本由 VALID 迁移到 PLATSUSPENDVALID 的状态应该恢复到 VALID

平台管理员需要只先暂停服务上才能取消服务商变成 PLATCANCEL 态，此时所有选择该服务商定义的服务的商户都不能再提供服务

(3) 平台管理员对 SHOPSUSPEND 商户服务的操作：

平台管理员暂停服务商，对于原本处在 SHOPSUSPEND 下商户服务将变为 PLATSUSPENDSUSPEND，若此时平台管理员选择恢复，必须将先前处在 SHOPSUSPEND 的状态恢复成 SHOPSUSPEND

这里设计还是需要平台管理员先去暂停才能取消，因此没有设计从 SHOPSUSPEND 到 PLATCANCEL，主要是希望保证业务逻辑上的相似性

(4) 设计注意点：

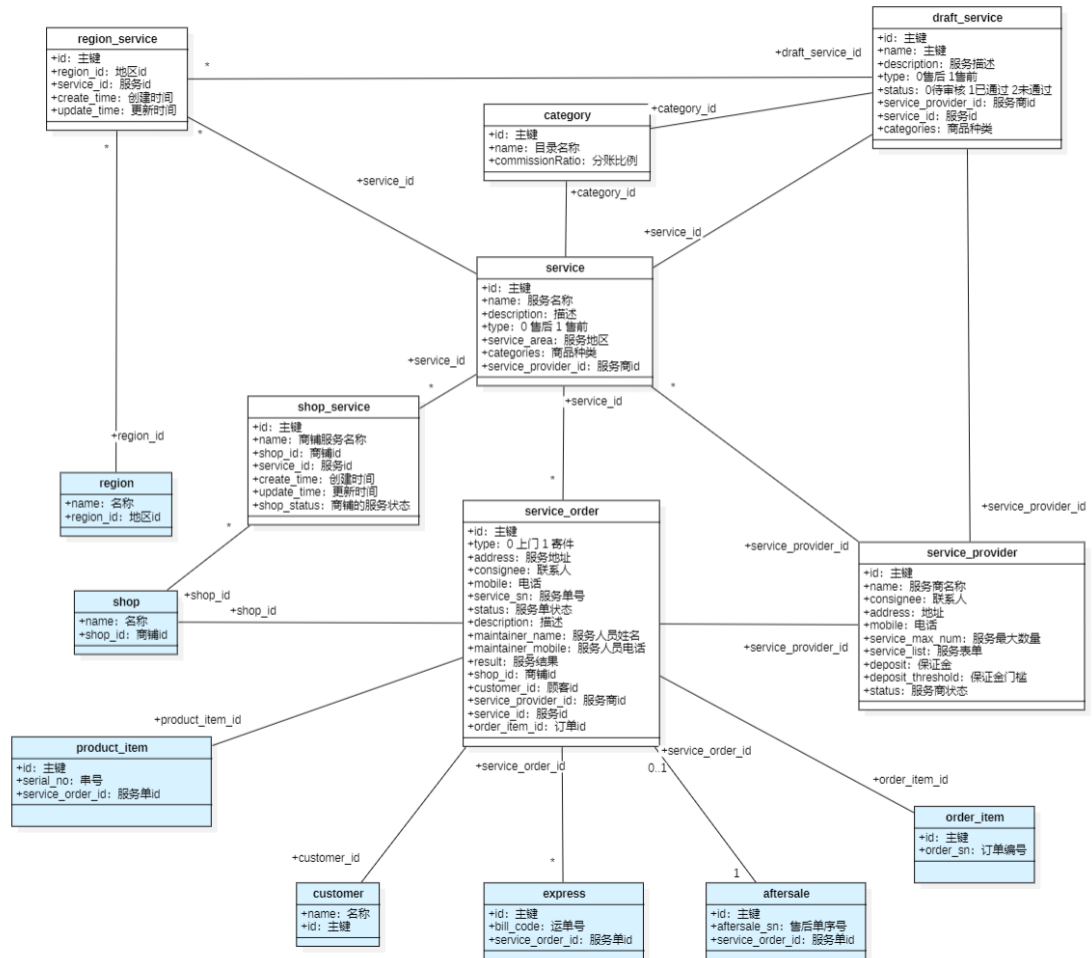
对于同一个恢复操作应该保证恢复到原有的状态，因此做 PLATSUSPENDSUSPEND 和 PLATSUSPENDVALID 的区分

商户权限应小于平台管理员的权限，例如保证平台管理员暂停服务商时，商户不能修改平台管理员的处理，而应该暂时不能使用该服务

服务 Service 简要说明

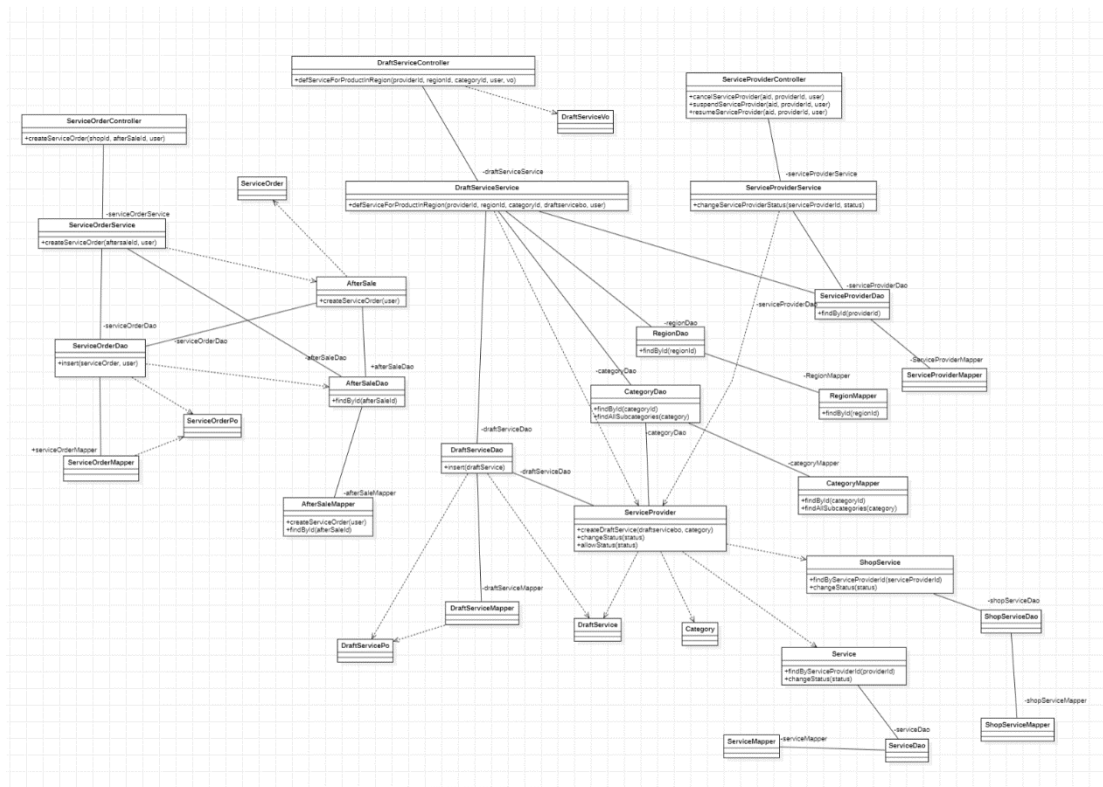
服务商定义的服务只需要关联到服务商的状态，若当前服务商正处在 VALID 或被平台管理员恢复为 VALID 才能提供服务，Service 有效；若当前服务商被取消或者暂停时，Service 都无效。因此只做两个状态的区分

4. 数据库



5. 程序逻辑

5.1 静态模型



5.2 服务商

5.2.1 服务商修改服务商状态

PUT
/adminusers/{aid}/maintainers/{mid}/suspend 平台管理员暂停服务商*

• 修改服务商状态为暂停

Parameters
Try it out

Name	Description
aid * required integer (path)	管理员id <input type="text" value="aid"/>
mid * required integer (path)	服务商id <input type="text" value="mid"/>
authorization * required string (header)	用户token <input type="text" value="authorization"/>

Responses
Response content type
application/json

Code	Description
default	成功

Example Value | Model




```

{
  "errno": 0,
  "errmsg": "成功"
}

```

PUT

/adminusers/{aid}/maintainers/{mid}/resume 平台管理员恢复服务商*

- 修改服务商状态为恢复服务

Parameters

Try it out

Name	Description
aid * required integer <i>(path)</i>	管理员id <input type="text" value="aid"/>
mid * required integer <i>(path)</i>	服务商id <input type="text" value="mid"/>
authorization * required string <i>(header)</i>	用户token <input type="text" value="authorization"/>

Responses

Response content type

application/json

Code	Description
default	成功

Example Value

Model

```
{  "errno": 0,  "errmsg": "成功"}
```

PUT

/adminusers/{aid}/maintainers/{mid}/cancel 平台管理员取消服务商*

修改服务商状态为取消

Try it out

Parameters

Name	Description
aid * required integer (path)	管理员id <input type="text" value="aid"/>
mid * required integer (path)	服务商id <input type="text" value="mid"/>
authorization * required string (header)	用户token <input type="text" value="authorization"/>

Responses

Response content type application/json

Code	Description
default	成功

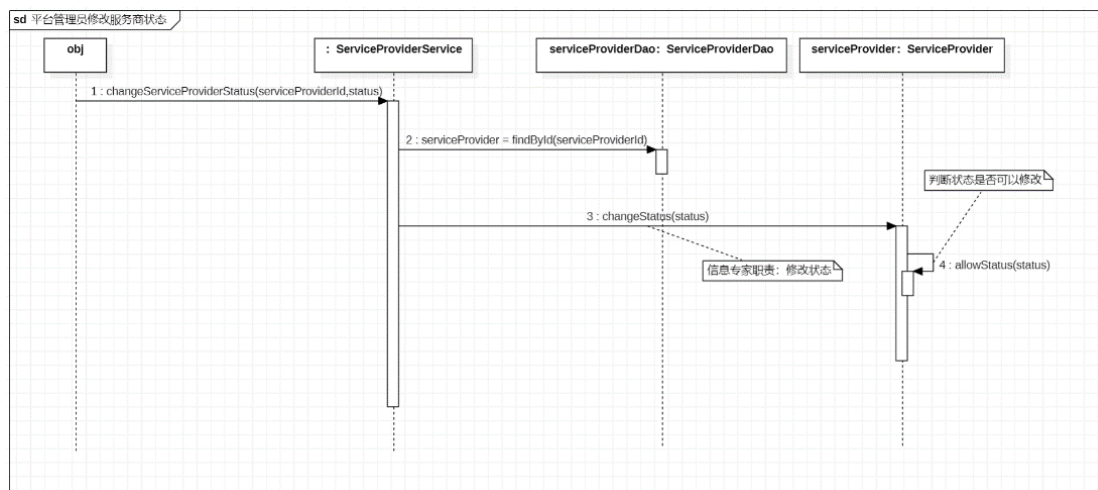
Example Value

Model

```

{
  "errno": 0,
  "errmsg": "成功"
}

```



说明：

将平台管理员暂停、恢复、取消服务商合并成一个方法为修改服务商状态，并在后端的 Service 层中使用函数 changeServiceProvider 实现。首先根据 API 路径上的 aid 作为函数参数的 serviceProviderId，在接收前端以上三个不同的 API 对应的 Controller 层会设定不同的 status 并同时作为参数，统一调用 Service 层的 changeServiceProvider。

在时序图中，首先根据 serviceProviderId 调用 serviceProviderDao.findById(service

ProviderId) 获得对应的服务商 serviceProvider。根据信息专家的原则，在 ServiceProviderService 中将修改状态 changeStatus(status) 的职责分配给 bo 对象 serviceProvider。

changeStatus(status) 里先使用 allow 方法，根据 ServiceProvider 状态机图判断当前服务商的状态 serviceProvider.status 是否可以迁移为 status：

如果可以迁移，则修改 serviceProvider.status= status

如果不可以迁移，则不允许修改，返回错误码 REFUSE_TO_CHANGE_STATUS

同时注意，服务商在不同状态下会有不同的行为限制。

5.3 服务

5.3.1 服务商定义在某个地区为某种商户的服务

POST

/maintainers/{mid}/region/{rid}/categories/{id}/service

服务商定义在某个地区为某种商品的服务*

📄 ↶ ↷

- 需要登录
- 默认为无效
- priority默认为1000

Parameters

Try it out

Name	Description
authorization * required string (header)	用户token <input type="text" value="authorization"/>
mid * required integer (\$int32) (path)	服务商id <input type="text" value="mid"/>
rid * required integer (path)	地区id <input type="text" value="rid"/>
id * required integer (\$int32) (path)	商品类别id <input type="text" value="id"/>
ServiceDraft * required object (body)	<div>Example Value Model</div> <div><pre>{ "name": "string", "description": "string", "type": 0 }</pre></div> <div>Parameter content type <div>application/json</div></div>

Responses Response content type application/json

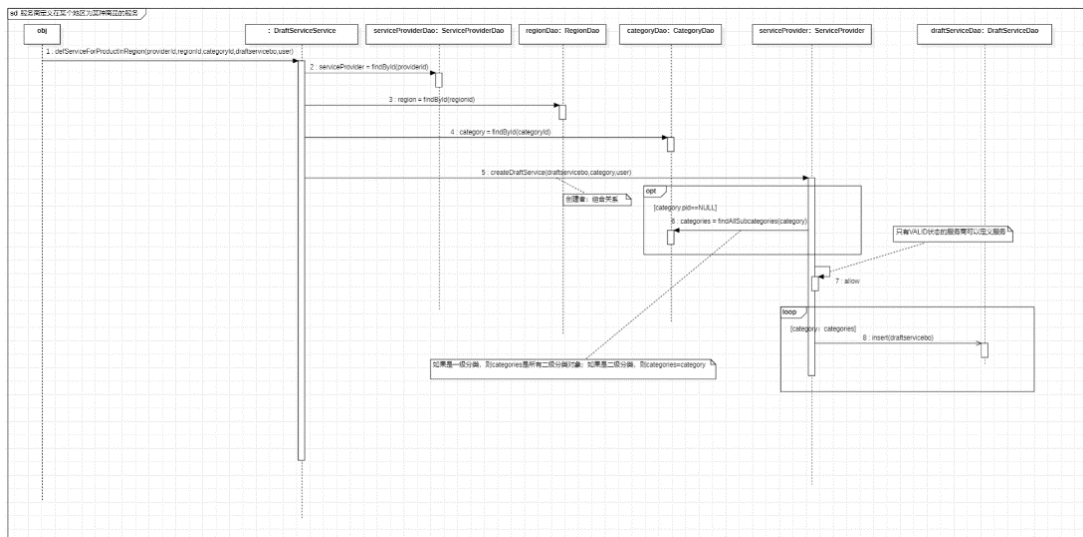
Code	Description
default	成功

Example Value | Model

```

{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 0,
    "name": "string",
    "description": "string",
    "type": 0,
    "service_area": {
      "id": 0,
      "name": "string"
    }
  }
}

```



说明:

服务商定义在某个地区为某种商品的服务，即定义 draftService 其主要原因是服务商定义的服务需要经过平台管理员的审核。API 路径上的 mid、rid、id 分别对应到函数 defServiceForProductInRegion 上的 providerId、regionId、categoryId。

首先根据 providerId 调用 serviceProviderDao 找到当前服务商 serviceProvider。在本模块的 Mapper 层使用 RestFul API 调用外部 region 模块，并在本模块 Dao 层次实现 RegionDao，调用 RegionDao 的 findById(regionId) 保证能找到 region。

由于服务跟服务商是组合关系，根据创建者原则，由服务商 serviceProvider 创建服务 service，将业务工作 createDraftService 放在 bo 对象 serviceProvider，满足六边型体系结构。此处注意将 category 传到函数 createDraftService 内，要能使服务商成功定义服务需要满足：若 category.pid==NULL，说明为一级分类，则调用外部 product 模块的 findAllSubcategories 找到所有二级分类；否则，categories 就是 category

根据状态机图，只有当前服务商状态为有效态 VALID，才能创建服务

使用 loop 循环，将所有分类 categories 里的 category 更新 draftService.categoryId，再将更新好的每个 draftService 都插入到数据库，此时数据库里记录的 draftService 里的类型字段是二级分类

5.4 服务单

5.4.1 创建服务单

POST

/internal/shops/{did}/aftersale/{aid}/serviceOrders 创建服务单

Try it out

Parameters

Name	Description
authorization * required string (header)	用户token <input type="text" value="authorization"/>
did * required integer (path)	商铺id <input type="text" value="did"/>
aid * required integer (path)	售后单id <input type="text" value="aid"/>

Responses

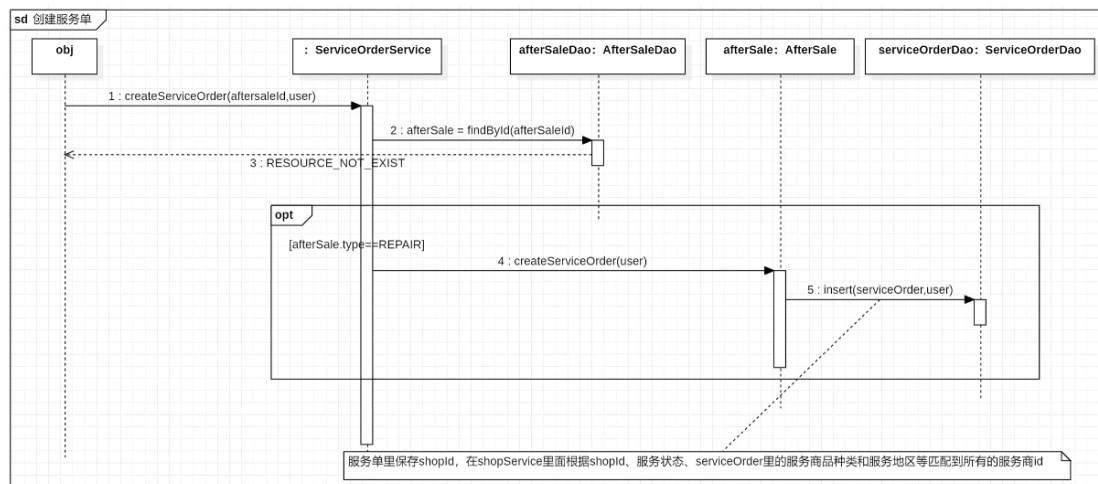
Response content type

application/json

Code	Description
default	成功

Example Value | Model

```
{
  "errno": 0,
  "errmsg": "成功",
  "data": {
    "id": 0,
    "type": 0,
    "address": "string",
    "consignee": "string",
    "mobile": "string",
    "serviceSn": 0,
    "status": "string",
    "description": "string",
    "maintainerName": "string",
    "maintainerMobile": "string",
    "result": "string",
    "shop_id": 0,
    "customer_id": 0,
    "service_provider_id": 0,
    "service_id": 0,
    "oeder_item_id": 0
  }
}
```



说明：

创建服务单时，在系统中目前只考虑从售后单创建服务单的业务，首先对于 API 上的 did 会使用 AOP 验证参数有效性，防止不存在的商铺。API 路径上的 aid 对应到函数 createServiceOrder 上的 afterSaleId，由于 AfterSale 属于外部模块，因此需要在本模块的 Mapper 和 Dao 层分别实现类 AfterSaleMapper 和 AfterSaleDao，并在 Mapper 里通过微服务的方式调用 Restful 发送 Http 请求 AfterSale 模块服务。

首先通过 findById 根据 afterSaleId 找到对应的 afterSale，若当前数据库不存在该 afterSale，则说明不存在售后单，则不能对一个不存在的售后单创建服务单，返回错误码 RESOURCE_NOT_EXIST

若存在售后单，还需要验证当前售后单是一个维修 REPAIR 类型的售后单才可以创建服务单。对于退货、换货等售后单不需要创建服务单。

根据创建者原则条件之一，若符合 B 有 A 的初始化数据则可以由 B 对象创建 A 对象。由于 API 是一个 internal 的 API，不能由系统外部访问，只能由系统内部调用，售后单里包含所有服务单里的数据，因此使用 afterSale 的 createServiceOrder 创建服务单，并使用 serviceDao 创建 serviceOrder 将其 insert 插到数据库

注意一个细节：在业务上，要确保满足服务单条件(服务商品类型、服务地区等) 并且商家选择的有效的服务商才可以看到该未被接单的服务单，提供给服务商接单，服务单里保存 shopId，而在对象模型中使用 ShopService 记录一个商家选择的服务商定义的所有服务，其中包括服务类型、服务地区、服务状态、服务商的 Id，只属于这个服务商 Id 的集合里的服务商才能查看该服务单并选择接单