**海尔大uHome云平台**

**通用SDK使用手册**

版本：V0.1

2015年05月

文档修订记录

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **版本编号** | **\*变化状态** | **变更内容和范围** | **变更日期** | **变更人** | **批准日期** | **批准人** |
| 0.1 | A | 初稿 | 2015-05-28 | 刘建锋 |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

\*变化状态：A——增加，M——修改，D——删除，N——正式发布

文档审批信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **审阅人** | **角色** | **审阅日期** | **签字** | **备注** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**目录**

[1 引言 4](#_Toc420677260)

[1.1 执行摘要 4](#_Toc420677261)

[1.2 编写目的 4](#_Toc420677262)

[1.3 术语和定义 4](#_Toc420677263)

[1.4 参考资料 5](#_Toc420677264)

[1.5 特殊记号格式说明 5](#_Toc420677265)

[2 系统说明 5](#_Toc420677266)

[3 主要接口说明 5](#_Toc420677267)

[3.1 用户系统 5](#_Toc420677268)

[3.1.1 UserServiceImpl 5](#_Toc420677269)

[3.1.1.1 register() 5](#_Toc420677270)

[3.1.1.2 query() 5](#_Toc420677271)

[3.1.1.3 updateUserProfile() 6](#_Toc420677272)

[3.1.1.4 resetPassword() 6](#_Toc420677273)

[3.1.2 UserExtendServiceImpl 6](#_Toc420677274)

[3.1.2.1 queryEmail() 6](#_Toc420677275)

[3.1.2.2 queryMobile() 6](#_Toc420677276)

[3.1.2.3 updateEmail() 7](#_Toc420677277)

[3.1.2.4 updateMobile() 7](#_Toc420677278)

[3.2 能力系统 8](#_Toc420677279)

[3.2.1 DeviceBindingServiceImpl 8](#_Toc420677280)

[3.2.1.1 Bind() 8](#_Toc420677281)

[3.2.1.2 Unbind() 8](#_Toc420677282)

[3.2.1.3 queryDevices() 8](#_Toc420677283)

[3.2.1.4 queryUsers() 9](#_Toc420677284)

[3.2.1.5 renameDevice() 9](#_Toc420677285)

[3.2.2 WeatherdubboServiceImpl 9](#_Toc420677286)

[3.2.2.1 currWeather() 9](#_Toc420677287)

[3.2.2.2 forecast7daysWeather() 9](#_Toc420677288)

[3.2.2.3 currWeatherExponent() 10](#_Toc420677289)

[3.2.2.4 currAirExponent() 10](#_Toc420677290)

[3.3 设备系统 10](#_Toc420677291)

[3.3.1 OperateServiceImpl 10](#_Toc420677292)

[3.3.1.1 deviceAttrOp() 10](#_Toc420677293)

[3.3.1.2 deviceGroupOp() 11](#_Toc420677294)

[3.3.1.3 deviceInfraredOp() 11](#_Toc420677295)

[3.3.1.4 deviceOpAllStatus() 12](#_Toc420677296)

[4 接口使用 13](#_Toc420677297)

[4.1 引入Spring 的工程中配置 13](#_Toc420677298)

[4.1.1 rest 方式请求接口类配置 13](#_Toc420677299)

[4.1.2 dubbo方式请求接口类配置 13](#_Toc420677300)

[4.2 一般工程中配置 14](#_Toc420677301)

[4.2.1 rest 方式请求接口类实现 14](#_Toc420677302)

[4.2.2 dubbo方式请求接口类实现 14](#_Toc420677303)

[4.3 openservice.properties说明 15](#_Toc420677304)

[5 示例代码 16](#_Toc420677305)

[5.1 接口类示例1 16](#_Toc420677306)

[5.2 接口类示例2 17](#_Toc420677307)

[6 常见问题 17](#_Toc420677308)

[6.1 NullPointerException 17](#_Toc420677309)

[6.2 UnknownHostException 18](#_Toc420677310)

[6.3 404 18](#_Toc420677311)

[6.4 Dubblo配置 18](#_Toc420677312)

# 引言

## 执行摘要

简述项目缘起、过程、结果以及其他需要重点强调的事项。本节内容相当于给管理人员的简介。

## 编写目的

本文档是关于海尔大uHome项目云平台的通用SDK使用手册，编写目的在于：帮助开发人员熟悉通用SDK（openservice-client）的使用，快速定位问题；解答通用SDK常见问题。

## 术语和定义

**智能家电**：指加装了WIFI联网模块、能够通过互联网远程控制的海尔家电设备，即海尔生产的带智能联网（WIFI）模块和远程程序控制模块的家电产品。有时，也称为物联网家电。目前包括冰箱、空调、洗衣机、酒柜、烤箱、热水器等类别。

**云平台**：海尔全集团范围的智能家电运营管理平台，是海尔集团互联网转型战略的重要组成之一。

## 参考资料

本设计所参考的相关文档、标准等资料如下：

* 云平台用户系统接口设计V1.2.1
* openservice-client-trunk接口使用说明
* M2M-APP\_M2M-Common-APP
* 能力化平台-服务接口说明

## 特殊记号格式说明

本文档暂未使用特殊文档记号或格式。

# 系统说明

        本SDK是为方便个应用系统调用平台相关服务而开发的工具，本SDK本身不进行任何业务处理，对应各服务的调用都是透传至后台系统。

# 主要接口说明

列举几个主要接口进行说明，其他接口参见《openservice-client-trunk接口使用说明》。

## 用户系统

### UserServiceImpl

接口类为UserService，所有接口都是REST方式

#### register()

|  |  |
| --- | --- |
| 方法功能： | 用户注册 |
| 方法参数： | ServiceContext ctx  UserAccount uacc |
| 参数说明： | uacc: 用户账号信息参照公共结构UserAccount |
| 返回参数： | Void |
| 参数说明： |  |

#### query()

|  |  |
| --- | --- |
| 方法名： | 查询用户信息 |
| 方法参数： | ServiceContext ctx  int accType  String userId  int idType |
| 参数说明： | accType: 账户类型  userId: 用户id  idType: ID类型0通过userId查询其他通过loginName查询 |
| 返回参数： | User |
| 参数说明： | 参照公共结构User |

#### updateUserProfile()

|  |  |
| --- | --- |
| 方法名： | 用户信息修改 |
| 方法参数： | ServiceContext ctx  String userId  Map<String, String> profile |
| 参数说明： | userId: 用户id  profile: 用户扩展信息 |
| 返回参数： | Void |
| 参数说明： |  |

#### resetPassword()

|  |  |
| --- | --- |
| 方法名： | 密码找回/重置 |
| 方法参数： | ServiceContext ctx  UserResetPasswordreq |
| 参数说明： | req: 用户信息及新密码参照公共结构UserResetPassword |
| 返回参数： | Void |
| 参数说明： |  |

### UserExtendServiceImpl

接口类为UserExtendService，所有接口都是REST方式

#### queryEmail()

|  |  |
| --- | --- |
| 方法名： | 获取用户邮箱(get)，屏蔽部分位 |
| 方法参数： | ServiceContext ctx  String userId |
| 参数说明： | userId: 用户id，系统生成的id非手机号 |
| 返回参数： | String |
| 参数说明： | 用户邮箱 |

#### queryMobile()

|  |  |
| --- | --- |
| 方法名： | 用户获取手机号(get)，屏蔽部分位 |
| 方法参数： | ServiceContext ctx  String userId |
| 参数说明： | userId: 用户id，系统生成的id非手机号 |
| 返回参数： | String |
| 参数说明： | 用户手机号 |

#### updateEmail()

|  |  |
| --- | --- |
| 方法名： | 用户修改邮箱  修改邮箱流程:  1、用户发起获取验证码  2、appServer调用UserService#genUVC(…)返回transactionId，同时平台将验证码发给用户  3、用户使用验证码进行验证并通过  4、appServer使用第2步返回的transactionId修改用户邮箱 |
| 方法参数： | ServiceContext ctx  String userId  String newEmail  String transactionId |
| 参数说明： | userId: 用户id，系统生成的id非手机号  newEmail: 用户新设置的邮箱地址  transactionId: 修改手机号的凭据 |
| 返回参数： | String |
| 参数说明： | 00000（正确码） |

#### updateMobile()

|  |  |
| --- | --- |
| 方法名： | 用户修改手机号  修改手机号流程:  1、用户发起获取验证码  2、appServer调用UserService#genUVC(…)返回transactionId，同时平台将验证码发给用户  3、用户使用验证码进行验证并通过  4、appServer使用第2步返回的transactionId修改用户手机号 |
| 方法参数： | ServiceContext ctx  String userId  String newMobile  String transactionId |
| 参数说明： | userId: 用户id，系统生成的id非手机号  newMobile: 用户新设置的手机号  transactionId: 修改手机号的凭据 |
| 返回参数： | String |
| 参数说明： | 00000（正确码） |

## 能力系统

### DeviceBindingServiceImpl

接口类为DeviceBindingService，所有接口都是REST方式

#### Bind()

|  |  |
| --- | --- |
| 方法名： | 用户绑定设备 |
| 方法参数： | ServiceContext ctx  String userId  Device[] devices |
| 参数说明： | userId: 用户id  devices: 设备列表参照公共结构Device |
| 返回参数： | String[] |
| 参数说明： | 未绑定成功设备mac列表 |

#### Unbind()

|  |  |
| --- | --- |
| 方法名： | 用户设备解除绑定 |
| 方法参数： | ServiceContext ctx  String ownerId  String deviceId  String[] userIds |
| 参数说明： | ownerId: 设备所有者userId  deviceId: 设备mac  userIds: 解绑用户ID列表 |
| 返回参数： | String[] |
| 参数说明： | 未解除绑定设备mac列表 |

#### queryDevices()

|  |  |
| --- | --- |
| 方法名： | 查询用户设备列表 |
| 方法参数： | ServiceContext ctx  String userId  String type  String subType  String specialCode  String typeIdentifier |
| 参数说明： | userId: 用户id  type: 设备大类  subType: 设备中类  specialCode: 设备专用号  typeIdentifier: 设备类型唯一识别码 |
| 返回参数： | Device[] |
| 参数说明： | 参照公共结构Device |

#### queryUsers()

|  |  |
| --- | --- |
| 方法名： | 查询用户设备上的所有用户信息 |
| 方法参数： | ServiceContext ctx  String deviceId |
| 参数说明： | deviceId: 设备mac |
| 返回参数： | UserBase[] |
| 参数说明： | 参照公共结构UserBase |

#### renameDevice()

|  |  |
| --- | --- |
| 方法名： | 设备重命名 |
| 方法参数： | ServiceContext ctx  String userId  String deviceId  String name |
| 参数说明： | userId: 用户id  deviceId:设备mac  name: 新名称 |
| 返回参数： | void |
| 参数说明： |  |

### WeatherdubboServiceImpl

接口类为WeatherService，所有接口都是dubbo方式

#### currWeather()

|  |  |
| --- | --- |
| 方法名： | 根据城市编码和语言获取当前天气 |
| 方法参数： | ServiceContext ctx  long areaid  String language |
| 参数说明： | areaid: 城市编码，不得为空  language: 语言，不得为空；zh\_CN表示中文 |
| 返回参数： | RealWeather |
| 参数说明： | 参照公共结构RealWeather |

#### forecast7daysWeather()

|  |  |
| --- | --- |
| 方法名： | 获取未来7天天气信息 |
| 方法参数： | ServiceContext ctx  long areaid  String language |
| 参数说明： | areaid: 城市编码，不得为空  language: 语言，不得为空；zh\_CN表示中文 |
| 返回参数： | ForecastWeather |
| 参数说明： | 参照公共结构ForecastWeather |

#### currWeatherExponent()

|  |  |
| --- | --- |
| 方法名： | 根据城市编码查询实时天气指数 |
| 方法参数： | ServiceContext ctx  long areaid |
| 参数说明： | areaid: 城市编码，不得为空 |
| 返回参数： | WeatherExponent[] |
| 参数说明： | [参照公共结构WeatherExponent](#_WeatherExponent) |

#### currAirExponent()

|  |  |
| --- | --- |
| 方法名： | 根据城市编码查询实时空气质量 |
| 方法参数： | ServiceContext ctx  long areaid |
| 参数说明： | areaid: 城市编码，不得为空 |
| 返回参数： | AirExponent |
| 参数说明： | [参照公共结构AirExponent](#_AirExponent) |

## 设备系统

### OperateServiceImpl

接口类为DeviceBindingService，所有接口都是REST方式

#### deviceAttrOp()

|  |  |
| --- | --- |
| 方法名： | 设备单操作，发送指令，需要配置异步应答 |
| 方法参数： | String systemId  long timestamp  String issuerId  String targetId  String name  String args  String sn |
| 参数说明： | systemId: 应用的systemId  timestamp: 操作发起时间，1970年开始的毫秒级  System.currentTimeMillis  issuerId: 用户登录后的accessToken  targetId: 设备mac地址  name: 指令6位码  args: BASE64编码例{"value": "操作值"}  sn: 操作的操作码，操作的异步应答使用此码标示。必须唯一。建议使  用手机mac+线程号+设备mac+System.currentTimeMillis |
| 返回参数： | String |
| 参数说明： | 00000（正确码） |

#### deviceGroupOp()

|  |  |
| --- | --- |
| 方法名： | 设备组操作，发送指令，需要配置异步应答 |
| 方法参数： | String systemId  long timestamp  String issuerId  String targetId  String name  String args  String sn |
| 参数说明： | systemId: 应用的systemId  timestamp: 操作发起时间，1970年开始的毫秒级  System.currentTimeMillis  issuerId: 用户登录后的accessToken  targetId: 设备mac地址  name: 组操作ID  args: BASE64编码例{"value":[{"name" : "\*\*\*", "value" : "\*\*\*"}, ...]}  sn: 操作的操作码，操作的异步应答使用此码标示。必须唯一。建议使  用手机mac+线程号+设备mac+System.currentTimeMillis |
| 返回参数： | String |
| 参数说明： | 00000（正确码） |

#### deviceInfraredOp()

|  |  |
| --- | --- |
| 方法名： | 设备红外操作，发送指令，需要配置异步应答 |
| 方法参数： | String systemId  long timestamp  String issuerId  String targetId  String args  String sn |
| 参数说明： | systemId: 应用的systemId  timestamp: 操作发起时间，1970年开始的毫秒级  System.currentTimeMillis  issuerId: 用户登录后的accessToken  targetId: 设备mac地址  args: BASE64编码例{"value": "红外数据BASE64"}  sn: 操作的操作码，操作的异步应答使用此码标示。必须唯一。建议使  用手机mac+线程号+设备mac+System.currentTimeMillis |
| 返回参数： | String |
| 参数说明： | 00000（正确码） |

#### deviceOpAllStatus()

|  |  |
| --- | --- |
| 方法名： | 查询设备状态，发送指令，需要配置异步应答 |
| 方法参数： | String systemId  long timestamp  String issuerId  String targetId  String sn |
| 参数说明： | systemId: 应用的systemId  timestamp: 操作发起时间，1970年开始的毫秒级  System.currentTimeMillis  issuerId: 用户登录后的accessToken  targetId: 设备mac地址  sn: 操作的操作码，操作的异步应答使用此码标示。必须唯一。建议使  用手机mac+线程号+设备mac+System.currentTimeMillis |
| 返回参数： | String |
| 参数说明： | 00000（正确码） |

# 接口使用

## 引入Spring 的工程中配置

如果工程中使用了spring则可以将类的初始化交给spring，配置如下：

### rest 方式请求接口类配置

---------------------------------------------------------------------------------

<util:properties id=*"openServiceConfig"* location=*"classpath:openservice.properties"*/>

<!— rest请求工具类配置 -->

<bean id=*"restClient"* class=*"com.hshbic.m2m.rest.impl.SimpleJsonRestClient"* init-method=*"init"*>

<property name=*"config"* ref=*"openServiceConfig"*/>

</bean>

<bean id=*"amsService"* name=*"amsService"* class=*"com.hshbic.openservice.client.impl.AMSServiceImpl"*>

<property name=*"config"* ref=*"openServiceConfig"*/>

<property name=*"restClient"* ref=*"restClient"*/>

</bean>

<!— 用户接口实现类配置 rest请求-->

<bean id=*"userService"* name=*"userService"* class=*"com.hshbic.openservice.client.impl.UserServiceImpl"*>

<property name=*"config"* ref=*"openServiceConfig"*/>

<property name=*"restClient"* ref=*"restClient"*/>

<property name=*"AMSService"* ref=*"amsService"*/>

</bean>

----------------------------------------------------------------------------------

Controller中配置

@Autowired

**private** UserService userService;

### dubbo方式请求接口类配置

分别列举了rest和dubbo请求接口类

--------------------------------------------------------------------------------

<!-- dubbo配置开始 -->

<dubbo:application name=*"hshbic-enabling-service"* />

<dubbo:registry protocol=*"zookeeper"* address=*"zk1.uhome.haier.net:2181,zk2.uhome.haier.net:2181,zk3.uhome.haier.net:2181"* check=*"false"* file=*"false"* />

<dubbo:protocol name=*"dubbo"* port=*"20880"* />

<!—配置调用类 -->

<dubbo:reference id=*"weatherdubboservice"* timeout=*"8000"* check=*"false"* protocol=*"dubbo"* interface=*"com.hshbic.enabling.home.WeatherService"*/>

<!-- dubbo配置结束 -->

<!— 天气接口实现类配置 dubbo请求-->

<bean id=*"weatherService"* name=*"weatherService"* class=*"com.hshbic.openservice.dubboClient.impl.WeatherdubboServiceImpl"*>

<property name=*"weatherdubboservice"* ref=*"weatherdubboservice"*></property>

</bean>

----------------------------------------------------------------------------------

Controller中配置

@Autowired

**private** WeatherService weatherService;

## 一般工程中配置

如果工程中没有使用spring则需要自己初始化各个类，具体方法如下：

### rest 方式请求接口类实现

Properties config = **new** Properties();

InputStream in = getClass().getResourceAsStream("/openservice.properties");

**try** {

config.load(in);

} **catch** (IOException e) {

e.printStackTrace();

}

//实例化restq请求工具类

SimpleJsonRestClient restClient = **new** SimpleJsonRestClient();

restClient.setConfig(config);

AMSService amsService = **new** AMSServiceImpl();

amsService.setConfig(config);

amsService.setRestClient(restClient);

//实例化用户接口实现类

UserService userService = **new** UserServiceImpl();

userService.setAMSService(amsService);

userService.setConfig(config);

userService.setRestClient(restClient);

### dubbo方式请求接口类实现

String address = "";

**public** com.hshbic.enabling.home.WeatherService *weatherdubbloService;*

address="zk1.uhome.haier.net:2181,zk2.uhome.haier.net:2181,zk3.uhome.haier.net:2181";

//address = "zookeeper://192.168.190.216:2181";

//url = config.getString("dubboUrl");

//url ="dubbo://192.168.190.187:20880";

ApplicationConfig application = **new** ApplicationConfig();

application.setName("hshbic-enabling-service");

// 连接注册中心配置

RegistryConfig registry = **new** RegistryConfig();

registry.setProtocol("zookeeper");

registry.setAddress(address);

// 引用远程服务

ReferenceConfig<WeatherService> reference = **new** ReferenceConfig<WeatherService>();

// 此实例很重，封装了与注册中心的连接以及

// 与提供者的连接，请自行缓存，否则可能造成内存和连接泄漏

reference.setApplication(application);

reference.setRegistry(registry); // 多个注册中心可以用setRegistries()

reference.setInterface(WeatherService.**class**);

reference.setProtocol("json");

reference.setProtocol("dubbo");

//reference.setUrl(url);

reference.setTimeout(8000);

// 和本地bean一样使用xxxService

*weatherdubbloService* = reference.get();

WeatherdubboServiceImpl *weatherService* = **new** WeatherdubboServiceImpl();

*weatherService*.setWeatherdubboservice(*weatherdubbloService*);

## openservice.properties说明

Rest 请求类中不同的接口会访问不同的服务地址,所以需要在openservice.properties中配置各服务地址和请求路径

**openservice.properties:**

#各服务地址配置

user.baseUrl = http://open.uhome.haier.net/uc/users

#security.baseUrl = http://uhome.haier.net/uc/auth

security.baseUrl = http://uhome.haier.net/security

security.login.baseUrl = http://uhome.haier.net/security

cif.baseUrl = http://open.uhome.haier.net/common

enabling.home.baseUrl = http://open.uhome.haier.net/enabling/home

enabling.baseUrl = http://open.uhome.haier.net/enabling/service

message.baseUrl = http://open.uhome.haier.net/enabling/service

syn.op.baseUrl = http://open.uhome.haier.net/commonOpResultService

ams.baseUrl = http://internal.uhome.haier.net/m2m-cloud-manager

domainModel.baseUrl = <http://internal.uhome.haier.net/model>

#各服务配置请求路径

#Query/Change User Email

user.queryEmail = /{userId}/email

#Query/Change User Mobile

user.queryMobile = /{userId}/mobile

#Change User Info

user.profile = /{userId}/userProfile

#Apply Verification Code

user.apply = /applyMsgCode

#Verify Verification Code

user.verify = /checkMsgCode

user.thirdUserProfile = /thirdUser

user.queryThirdUserProfile = /getUserbyThirdName

user.queryUserList = /{userId}/queryUserList

user.userInfoByAttrib = /userInfoByAttrib

user.profileValidate = /profileValidate

user.primeValidate = /primeValidate

user.tenantUsers = /tenantUsers

user.getBindStatus = /getBindStatus

user.bindExistAcct = /bindExistAcct

user.unBindUser = /unBindUser

user.queryThirdUserbyUserIds = /{accType}/queryThirdUserbyUserIds

user.saveMsgstatus = /saveMsgStatus

user.applyLoginCode = /applyLoginCode

user.pwdReset = /pwdReset

user.pwdFind = /pwdFind

cif.device.bind = /newusers/{userId}/devices

cif.device.unbind = /devices/{deviceId}/bindingUsers

cif.device.rename = /users/{userId}/devices/{deviceId}/name

cif.device.updateInfo = /devices/{deviceId}/profile

cif.device.operate = /devices/{deviceId}/op/{sn}

cif.device.history = /devices/history/statuses

cif.device.statuses.online = /devices/{deviceId}/statuses/online

cif.device.dataReportStatus = /devices/{deviceId}/dataReportStatus

# 示例代码

## 接口类示例1

UserPwd userPwd = **new** UserPwd();

userPwd.setLoginType(userPwdRequest.getLoginType());

userPwd.setAttrValue(userPwdRequest.getAttrValue());

userPwd.setAccType("0");

**try** {

ServiceContext ctx = **new** ServiceContext(appId, appVersion,

clientId);

*logger*.debug("[pwdFind]ctx=[appId=" + appId + ",appVersion=" + appVersion + ",clientId=" + clientId + "]");

*logger*.debug("[pwdFind]userPwd=" + userPwd);

**long** st = System.*currentTimeMillis*();

userService.pwdFind(ctx, userPwd);

**long** et = System.*currentTimeMillis*();

*logger*.debug("[pwdFind] weatherService.pwdFind take " + (et - st) + " ms");

resp.save(ServiceErrorCode.*OK*, "操作成功");

} **catch** (ServiceException e) {

*logger*.error(e.toString(),e);

resp.save(e.getErrorCodeStr(), e.getMessage());

**return** resp;

} **catch** (Exception e) {

*logger*.error(e.toString(),e);

resp.save(CommonRestResponse.*ERROR\_SYSTEM*, "系统错误");

**return** resp;

}

## 接口类示例2

HashMap<String,Object> map = **new** HashMap<String,Object>();

ServiceContext ctx = **new** ServiceContext(appId, appVersion, clientId);

**try**{

*logger*.debug("[meteorologicalIndex]ctx=[appId=" + appId + ",appVersion=" + appVersion + ",clientId=" + clientId + "]");

*logger*.debug("[meteorologicalIndex]areaid=" + areaid);

**long** st = System.*currentTimeMillis*();

WeatherExponent[] indexs = weatherService.currWeatherExponent(ctx, areaid);

**long** et = System.*currentTimeMillis*();

*logger*.debug("[meteorologicalIndex] weatherService.currWeatherExponent take " + (et - st) + " ms");

*logger*.debug("[meteorologicalIndex]indexs=" + indexs);

map.put("retCode", ErrorCode.*CODE\_00000*);

map.put("retInfo", ErrorCode.*INFO\_00000*);

map.put("indexes", indexs);

**return** map;

}**catch**(BaseException e){

*logger*.error("", e);

map.put("retCode", e.getResCode());

map.put("retInfo", e.getResInfo());

**return** map;

}

# 常见问题

## NullPointerException

如果调用接口时抛出空指针异常，一般情况下是接口类中参数没有实例化，例如:

UserServiceImpl 中需要设置参数

userService.setAMSService(amsService);

userService.setConfig(config);

userService.setRestClient(restClient);

如果没有设置就有可能出现空指针异常，所以出现空指针异常时首先要检查接口类中参数是否配置。

## UnknownHostException

例如：

java.net.UnknownHostException: internal.uhome.haier.net

首选检查openservice.properties中的url是否配置正确，internal.uhome.haier.net是否是真正要请求的地址，如果检查没有问题那就是域名解析有问题，需要检查服务器配置。

## 404

如果调用接口时返回类似如下404错误，请检查openservice.properties中的访问路径是否正确。

{"retInfo":"Http Status code: 404","retCode":"00010","data":"{}"}

## Dubblo配置

采用dubblo连接的接口类配置时需要注意：

目前开发环境服务消费者需要绕过注册中心，采用点对点直连方式连接：

<dubbo:reference id=*"weatherdubboservice"* timeout=*"8000"* check=*"false"*

protocol=*"dubbo"* interface=*"com.hshbic.enabling.home.WeatherService"* url=*"dubbo://192.168.190.187:20880"* />

其余环境配置如下：

<dubbo:reference id=*"weatherdubboservice"* timeout=*"8000"* check=*"false"*

protocol=*"dubbo"* interface=*"com.hshbic.enabling.home.WeatherService"* />