应用平台

SDK 参考

文档版本 04

发布日期 2025-02-17

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 SDK 概述

AppStage为开发者提供运维中心SDK和AI原生应用引擎SDK，方便开发者将AppStage的能力快速集成到自己的应用中。

本文介绍了AppStage提供的开放API的SDK语言版本，列举了最新版本SDK的获取地址。

约束与注意事项

- 运维中心提供的SDK是基于Java1.8版本开发的，如果Spring Cloud项目使用Java11及以上版本，则不支持使用运维中心提供的SDK进行应用开发。
- AI原生应用引擎提供了Python语言的SDK，支持Python >= 3.10版本。

运维中心 SDK 类型介绍

运维中心为开发者提供访问凭证管理服务（Access Credential Management Service，简称ACMS）、服务发现（NUWA Cloud Map，简称Cloud Map）、负载均衡（Software/Server Load Balancer，简称SLB）和数据库治理（WiseDBA）的SDK，方便开发者将运维中心的能力快速集成到自己的应用中。

表 1-1 SDK 列表

SDK分类	说明	编程语言	下载地址	参考文档
ACMS的SDK：STS SDK	STS SDK（Security Token Service，简称STS）服务软件开发工具包是对访问凭据管理服务（ACMS）提供的REST API进行的封装，以简化用户的开发工作。用户直接调用SecurityTokenService SDK提供的接口函数即可实现使用ACMS业务能力的目的。 使用STS SDK即可使用运维中心ACMS的敏感配置项托管和微服务之间请求认证功能。	Java	• **nuwa-open-sdk-1.1.0-20240204093135.zip** • 完整性校验**nuwa-open-sdk-1.1.0-20240204093135.zip.sha256** 下载包nuwa-open-sdk-1.1.0-20240204093135.zip中的文件介绍请参见**表1-2**。	**STS SDK**
Cloud Map的SDK：Cloud Map SDK	Cloud Map SDK服务软件开发工具包是对服务发现（Cloud Map）服务提供的REST API进行的封装，以简化用户的开发工作。用户直接调用Cloud Map SDK提供的接口函数即可实现使用Cloud Map服务业务能力的目的。	Java		**Cloud Map SDK**
SLB的SDK：Gray SDK	Gray SDK负责业务的负载均衡，Nginx组件通过反向代理实现了业务的负载均衡，通过丰富的扩展功能，可以对HTTP消息定制丰富的控制策略。 使用Gray SDK即可使用运维中心负载均衡（SLB）的灰度管理功能。	Java		**Gray SDK**

SDK分类	说明	编程语言	下载地址	参考文档
WiseDBA的SDK：Rainbow SDK	Rainbow SDK构建云原生DevOps全流程可信build-in的数据库治理解决方案。使用Rainbow SDK即可使用运维中心的数据库治理（WiseDBA）功能。 • 提供MySQL/Cassandra/GaussDB/DRDS全流程设计、开发、发布、运维（管理、治理、诊断）方案。 • 可信build-in：过程可信，结果可信，接入安全（无人工接入密码），操作（资源高危操作）安全。	Java		Rainbow SDK

表 1-2 SDK jar 包介绍

文件夹	SDK分类	对应的jar包
nuwa-cse-sdk：适用于基于NUWA框架且不使用spring boot的项目	STS SDK	• sts-key-sdk-1.1.17.109.jar • sts-sdk-base-1.1.17.109.jar • cloudsoa-security-1.1.13.100.jar
	Cloud Map SDK	• nuwa-cloudmap-config-servicecomb-1.0.12.100.jar • nuwa-cloudmap-core-1.0.12.100.jar • nuwa-cloudmap-registry-cse3-1.0.12.100.jar • nuwa-cloudmap-registry-servicecomb-common-1.0.12.100.jar

文件夹	SDK分类	对应的jar包
	Gray SDK	• graysdk-core-1.4.14.500.jar • graysdk-cse3-1.4.14.500.jar • nuwa-apaas-graysdk-3.1.5.100.jar
	Rainbow SDK	• rainbow-proxy-1.2.18.102.jar • gpaas-middleware-common-2.2.6.100.jar
	NUWA基础SDK	• nuwa-boot-container-3.1.5.100.jar • nuwa-core-3.1.5.100.jar • nuwa-cse-foundation-3.1.5.100.jar • nuwa-share-libs-3.1.5.100.jar • nuwa-tenant-sdk-3.1.5.100.jar
spring-cloud-sdk：适用于基于Spring Cloud框架的项目	STS SDK	• sts-key-sdk-1.1.19.100.jar • sts-sdk-base-1.1.19.100.jar • sts-spring-boot-1.1.19.100.jar • cloudsoa-security-1.1.14.101.jar
	Cloud Map SDK	• nuwa-cloudmap-core-1.0.12.100.jar • nuwa-cloudmap-spring-boot-starter-1.0.12.100.jar • spring-cloud-starter-cloudmap-discovery-1.0.12.100.jar
	Gray SDK	不支持

文件夹	SDK分类	对应的jar包
	Rainbow SDK	• gpaas-jetcd-api-2.0.0.106.jar • gpaas-jetcd-v2-2.0.0.106.jar • gpaas-middleware-common-2.2.6.100.jar • nuwa-gpaas-rainbowproxy-3.2.2.100.jar • nuwa-rainbowproxy-spring-boot-starter-3.2.2.100.jar • rainbow-api-1.2.18.201.jar • rainbow-core-1.2.18.201.jar • rainbow-core-drds-1.2.18.201.jar • rainbow-core-gaussdb-1.2.18.201.jar • rainbow-proxy-1.2.18.201.jar
	NUWA基础SDK	• nuwa-boot-container-3.2.2.100.jar • nuwa-core-3.2.2.100.jar • nuwa-core-spring-boot-starter-3.2.2.100.jar
spring-cse-sdk：适用于基于NUWA框架且使用spring boot的项目	STS SDK	• sts-key-sdk-1.1.19.100.jar • sts-sdk-base-1.1.19.100.jar • sts-spring-boot-1.1.19.100.jar • cloudsoa-security-1.1.14.101.jar

文件夹	SDK分类	对应的jar包
	Cloud Map SDK	• nuwa-cloudmap-config-servicecomb-1.0.12.100.jar • nuwa-cloudmap-core-1.0.12.100.jar • nuwa-cloudmap-registry-cse3-1.0.12.100.jar • nuwa-cloudmap-registry-servicecomb-common-1.0.12.100.jar • nuwa-cloudmap-spring-boot-starter-3.2.2.100.jar • nuwa-gpaas-cloudmap-3.2.2.100.jar
	Gray SDK	• graysdk-core-1.4.14.500.jar • graysdk-cse3-1.4.14.500.jar • nuwa-apaas-graysdk-3.2.2.100.jar • nuwa-graysdk-spring-boot-starter-3.2.2.100.jar
	Rainbow SDK	• nuwa-gpaas-rainbowproxy-3.2.2.100.jar • nuwa-rainbowproxy-spring-boot-starter-3.2.2.100.jar • rainbow-proxy-1.2.18.102.jar • gpaas-middleware-common-2.2.6.100.jar

文件夹	SDK分类	对应的jar包
	NUWA基础SDK	• nuwa-boot-container-3.2.2.100.jar • nuwa-core-3.2.2.100.jar • nuwa-core-spring-boot-starter-3.2.2.100.jar • nuwa-cse-foundation-3.2.2.100.jar • nuwa-cse-foundation-spring-boot-starter-3.2.2.100.jar • nuwa-tenant-sdk-3.2.2.100.jar
tools：configparser为自定义参数解析工具，通过NUWA部署时，解析参数模板，将模板中的参数变量，替换为实际的配置项值，具体使用方式请参见**使用configparser工具优化代码**。	-	

AI 原生应用引擎 SDK 介绍

AI原生应用引擎面向开发者提供了一套搭建原生应用的Python SDK，包含了模型调用，知识获取，工具调用等功能。开发者可以使用SDK调用AI原生应用引擎的各种能力，快速构建大模型应用。

- AI原生应用引擎SDK获取地址：**wiseagent-dev-sdk-python**。您也可以参考**下载SDK**获取AI原生应用引擎SDK并进行完整性校验。
- AI原生应用引擎SDK参考文档：**AI原生应用引擎SDK API**。

2 SDK 功能介绍

运维中心 SDK 功能介绍

表 2-1 STS SDK 功能矩阵

功能	Java
读取微服务身份证书	√
连接STS-Server获取密钥、认证凭据等	√
连接STS-Server获取敏感配置	√
解密敏感数据	√
微服务间通信认证	√
微服务间通信敏感数据加密	√

表 2-2 Cloud Map SDK 功能矩阵

功能	Java
微服务注册发现	√
中间件注册发现	√
URL注册发现	√
支持CSE平滑切换到Cloud Map，支持cse访问不同clusterName的微服务	√
支持URL AZ内就近访问，指定灰度、Tag访问	√
支持获取所有的服务实例以及实例变更通知	√
提供restful调用的API	√
Cloud Map支持Spring Cloud	√

功能	Java
支持多个ELB/SLB，避免单点故障	√

表 2-3 Gray SDK 功能矩阵

功能	Java
高性能灰度规则执行	√
兼容CSE的灰度规则能力	√
支持订阅多个微服务灰度配置和执行	√
支持多阶段灰度发布	√
支持灰度生命周期管理	√
支持灰度规则执行异常检测	√

表 2-4 Rainbow SDK 功能矩阵

功能	Java
支持MySQL/Cassandra/GaussDB/DRDS	√
数据库主备动态切换	√
读节点发生故障，单线程读重试	√
基于集中式数据源信息管理和动态变更	√
基于JDBC规范，很容易扩展支持实现JDBC规范的数据源	√
应用直连数据库，无代理	√
平滑加减数据库节点	√
提供hint方式或者是编码式强制某条SQL读取从库或者主库等	√

AI 原生应用引擎 SDK 功能介绍

表 2-5 AI 原生应用引擎 SDK 功能矩阵

接口类型	接口名	函数名	Python
大模型相关接口	文本生成	ModelRouter.chat.completion.create	√

接口类型	接口名	函数名	Python
	向量生成	ModelRouter.embeddings.create	√
	文生图	ModelRouter.images.generate	√
	图生文	ModelRouter.images.image2text	√
知识库相关接口	知识库查询	KnowledgeRetriever.retrieve	√
	Filter构造	KnowledgeFilterCondition.single_filter_condition	√
	Filter融合	KnowledgeFilterCondition.combine_filter_condition	√
	Order构造	KnowledgeSortCondition.single_order_condition	√
	Order融合	KnowledgeSortCondition.combine_order_condition	√
Prompt相关接口	生成prompt模板	PromptTemplate.from_template	√
	实例化prompt模板	PromptTemplate.format	√
应用相关接口	调用应用	ApplicationCenter.get_tools	√

3 运维中心 SDK 介绍

3.1 STS SDK

3.1.1 概述

STS SDK服务软件开发工具包是对AppStage运维中心访问凭据管理服务（ACMS）提供的REST API进行的封装，以简化用户的开发工作。

STS SDK封装了业务微服务读取ACMS身份证书、到ACMS-Server上获取密钥、认证凭据、解密敏感数据、微服务间通信认证加密等功能，用户直接调用STS SDK提供的接口函数即可实现使用ACMS业务能力的目的。

3.1.2 使用 STS SDK（Spring Cloud 框架）

引入 STS SDK

在pom.xml中添加STS SDK依赖。

- 将${sts.version}替换成实际所使用的STS SDK版本。
- 如果将SDK放到外部maven仓中，则只需要添加sts-spring-boot依赖。
- 如果采用本地依赖的方式引入SDK，即手动将本地下载的SDK jar包引入到工程的lib目录下，还需要添加间接依赖：sts-key-sdk、sts-sdk-base、cloudsoa-security。

```
<dependency>
    <groupId>com.huawei.wisecloud.sts</groupId>
    <artifactId>sts-spring-boot</artifactId>
    <version>${sts.version}</version>
</dependency>
```

配置 STS

在微服务的ClassPath下增加添加配置文件sts/sts.properties（该文件路径可以通过环境变量sts.properties进行修改），内容为：

```
sts.server.domain=10.202.251.196:8080                                    #STS服务器的地址
sts.config.path=/opt/huawei/certs/xxxService/xxxMicroService/xxxMicroService.ini  #STS微服务证书路径，基础设施即代码（Infrastructure as Code，简称IaC）会将证书放在固定路径下，格式为/opt/huawei/certs/服务名/微服务名/微服务名.ini
```

初始化 STS

在启动类中增加注解@EnableStsAutoInitialization(value = "sts.properties")，注解的含义是启动STS自动初始化，其中value是指定STS的配置文件路径。

使用注解时，是通过加装Bean的方式初始化STS和解密敏感配置项，由于Bean的加载顺序不固定，有时会出现使用STS解密的代码被加载了，初始化STS的Bean还没有被加载。如果出现这种情况，可以在使用STS的类方法上添加如下方法解决此问题：

@Import(value = {StsEncryptablePropertiesConfiguration.class})

3.1.3 使用 STS SDK（NUWA 框架）

初始化 STS

NUWA中已经自带了STS插件，只需要在nuwa-module-config.yml文件中进行如下配置，即可初始化STS。这种方式可以保证在其他中间件、Cloud Map之前初始化STS，保证组件启动顺序正确。

```
nuwa:
  security:
    sts:
      enable: true
      serverDomain: xx.xx:xx
      configPath: /opt/huawei/certs/XXXService/XXXMicroService/XXXMicroService.ini
```

表 3-1 配置说明

参数	说明
nuwa.security.sts.enable	是否初始化STS，需要配置为true。
nuwa.security.sts.serverDomain	STS服务器的地址。
nuwa.security.sts.configPath	STS微服务证书路径，基础设施即代码（Infrastructure as Code，简称IaC）会将证书放在固定路径下，格式为/opt/huawei/certs/服务名/微服务名/微服务名.ini。

3.1.4 常见问题

- Domain not registered或Service not registered

  领域或微服务没有进行注册，请根据STS 2.0业务接入指南的指导自行注册领域或微服务。

- connect timed out或read timed out

  它有两种原因，connect timed out或read timed out:

  可以在sts.properties中添加超时配置，单位为毫秒：

  ```
  sts.connect.timeout=10000 //遇到 connect timed outsts.socket.timeout=10000 //遇到 read timed out
  ```

- Encrypt Permission Denied

  这是因为管理台上注册微服务时，默认将微服务注册成了NORMAL。只有角色为PLATFORM的微服务才有权限调用加密接口。

- fail to init session key

  fail to init session key

  fail to get master key from sts

  这是因为集成了STS 1.0，也就是集成了sts-cse-sdk，从STS 1.0的服务器获取master key失败。因为STS 1.0的服务器需要业务自己去搭建，在没有搭建的情况下，当然会获取失败。

  如果已经搭建，检查业务的yaml文件中配置的environment参数和STS 1.0微服务中的environment是否一致，如果不一致则修改成一致。

- mac check in GCM failed

  这个错误的原因就是密钥和密文不匹配导致无法解密。需要确认业务配置文件中的密文是从哪里来的?

  a. 通过IaC由配置中心下发：

     检查IaC代码中配置的敏感配置项ID中的服务或微服务名是否属于该运行的微服务。

     - 服务级：Service/{ServiceName}/{SensitiveName}/{tag}
     - 微服务级：MicroService/{ServiceName}/{MicroServiceName}/{SensitiveName}/{tag}

  b. 通过StsAgent加密后，手工配置：

     检查加密的时候使用的服务级别servicekek还是微服务级别kek，和密文中的级别是否一致，服务名和微服务名是否和该运行的微服务一致？加密得到密文的STS环境和运行的STS环境是否一致？例如测试环境加密的密文是不能在生产环境解密。

     ```
     # 使用微服务kek加密
     # /opt/huawei/apps/wisesecurity/stsagent/stsagent encrypt -s ServiceName -m
     MicroServiceName -x -h 10.33.102.162:8080
     # 使用服务servicekek加密
     # /opt/huawei/apps/wisesecurity/stsagent/stsagent encrypt -s ServiceName -m
     MicroServiceName -ts ServiceName -x -h 10.33.102.162:8080
     ```

  c. 中间件的口令解密

     - 业务集成了Cloud Map对接中间件，业务不用关心口令解密的问题。
     - 业务使用sts agent加密明文口令后配置到相应管理台。参考**2**，看加密用的密钥和代码中解密的密钥是否对应（服务级还是微服务级kek），以及加密时使用的微服务名和服务名是否一致。

  d. 不同微服务的证书不可混用，业务使用自己微服务的证书及配置文件进行SDK的初始化，解密自己微服务的配置。

- Undefined provider local

  当创建AESCryptor时，如果不用@指定KeyProvider，那么它会默认调用local KeyProvider。在使用kek或servicekek时，要指定KeyProvider为@sts。

  ```
  //错误
  AESCryptor cryptor = new AESCryptor.Builder().withKey("kek").withAlg(CryptoAlg.AES_GCM).build();
  //正确
  AESCryptor cryptor = new
  AESCryptor.Builder().withKey("kek@sts").withAlg(CryptoAlg.AES_GCM).build();
  ```

- 申请证书时Make sure haveged is running步骤或try active service步骤报错。

  这是因为机器上没有启动haveged服务。haveged是一个随机数的熵的提供方，它可以解决在某些情况下，系统熵过低的问题。

规避办法：在部署步骤前面加上sudo步骤；或者，到机器上，手动执行service haveged start命令，启动haveged服务。

解决方案：在申请证书步骤前添加sudo和shell-exec步骤，确保haveged服务为启动状态。

sudo systemctl enable haveged.service && sudo systemctl start haveged.service非EulerOS可以使用sudo service haveged start命令。

- service ini config file not exists

  这是因为STS在初始化时，找不到初始化配置文件（ini 文件）或者微服务名称.sts.p12证书文件不存在。该文件是在部署微服务过程中，通过stsagent申请证书时生成的，默认路径在/opt/huawei/certs/{serviceName}/{microServiceName}，业务也可以通过相应参数来修改。

  解决办法：

  a. 首先确认配置的STS参数：

     - 通过配置文件配置的，查看sts/sts.properties 文件，检查sts.config.path配置的文件是否存在。
     - 通过NUWA配置文件，检查microservice.yaml文件中nuwa.security.sts.configPath配置的文件是否存在。

       nuwa:security:sts:serverDomain: 10.33.102.109:8080configPath: /opt/huawei/certs/{ServiceName}/{MicroServiceName}/{MicroServiceName}.ini
     - 通过Java Bean方式配置的，也就是通过类似以下代码初始化，检查sts.config.path配置的文件是否存在

       Properties properties = new Properties();properties.setProperty("sts.server.domain", "10.33.102.162:8080");properties.setProperty("sts.config.path", "D:/Test/ConsumerService/ConsumerMicroService1/ConsumerMicroService1.ini");StsKeyApi.initWith(properties);
     - 如果配置的是相对路径，则必须是相对于resource目录的路径。如果是在IDEA中运行使用，则检查target下是否有该文件，如果没有则重新compiler，IDEA最终是从target下获取resource文件。

  b. 另外，请确认部署过程中证书申请的步骤在业务安装和启动流程之前，如果业务先启动，之后才进行证书的申请，配置文件还未生成，也会出现找不到配置文件的问题。

     若不存在，则要参考STS 2.0业务接入指南申请证书。如果配置的文件路径不是stsagent生成的默认路径，则需要手工将** /opt/huawei/certs/{ServiceName}/{MicroServiceName}/**拷贝到相应的路径下。

- Fail to derive key with master key

  这是因为STS没有初始化，就去调用了获取认证凭据的接口。

  解决办法：检查是否调用了StsKeyApi.init()或StsKeyApi.initWith(properties)。

- 如何解密微服务证书的口令

  微服务证书的口令使用本地随机的根密钥和工作密钥加密，口令明文使用StsKeyStoreUtil.getKeyStoreValue()进行解密。

- Access Permission Denied

  微服务集成STS SDK，获取相应微服务的Credentials时报如下错误Reqeust sts server fail, Request Token form xxxx&yyyy failed， Access Permission

Denied。 这是由于provider没有该微服务配置相应的ACL（Access Control List），导致没有权限获取访问该provider的credentials。

- Invalid api name

  STS Agent执行的时候报Invalid api name，这是由于 IaC 中使用了STS Agent高版本能力，但主机上STS Agent版本过低并且IaC中没有强制升级STS Agent导致的。通过IaC或手动更新STS Agent，使用1.1.5.100及之前的部署包时请打开force_install开关。

- 微服务有测试，镜像，海外，灰度等不同的环境，STS管理台如何管理这些敏感配置项?

  STS管理台在不同站点是分开部署。同一站点下假如想区分不同的场景，例如生产和灰度，开发，测试和镜像，则可以使用敏感配置项标签区分。

- STS管理台敏感配置项最大长度是多少?

  5000个字符。

- 本地缓存的STS KEK，ServiceKEK以及认证凭据是如何保护?
  - key.json由本地随机生成的rootkey和workkey加密保护，kek.json由STS KEK保护。
  - 加密算法采用AES GCM。

- 部署过程report certificate to cms报错，提示"An unknown error occurred when report certificates!"。

  首先检查/opt/huawei/certs/{service}/{microservice}下stscerttool.log中是否有错误信息。

  若日志中出现异常提示[Errno -2] Name or service not known，请使用以下命令确认系统版本是否为欧拉OS 2.9 [EulerOS release 2.0 (SP9)]：

  ```
  lsb_release -a
  ```

  a. 检查主机hostname是否超过64个字符，如果是的话，修改主机名。

  b. 主机存在无法解析出主机IP的问题，参考如下步骤检查：

     i. 检查/etc/nsswitch.conf中是否包含hosts项，且对应配置包含files或myhostname。

     ii. 若添加配置files，检查/etc/hosts，添加当前主机的IP映射[host_ip] [hostname]，例如：10.33.100.100 host-10-33-100-100。

     若添加配置myhostname，通常无需进行其他修改。

     iii. 运行以下命令检查是否有其他异常。

     ```
     hostname -i
     python -c "import socket;
     print(socket.gethostbyname(socket.getfqdn(socket.gethostname())))"
     ```

- 如何手工注册微服务

  使用IaC部署在WiseDBA上创建数据库实例，报没有注册STS的错误，如果业务还没有接入STS，可以在STS管理台进行手工注册该微服务。

- 如何将服务器上生成的STS微服务证书信息迁移到window机器上调试。

  将服务器上/opt/huawei/certs/{serviceName}/{microServiceName}目录下的所有问题打包下载到本机。修改{microServiceName}.ini 文件中 a_file，b_file ，c_file，d_file 的路径，该路径支持绝对路径或相对路径（相对于ini文件的路径，一般为{microServiceName}/apple/a ）。修改sts.properties中 sts.config.path为实际的ini文件的路径。

- missed = in sensitive string value / missed value in sensitive string

解密敏感配置项的时候报missed = in sensitive string value错误，这是由于敏感配置密文格式不正确。敏感配置项密文格式如下：

- kek加密密文格式： ENC(key=kek, value=xxxx)
- servicekek加密密文格式： ENC(key=servicekek, value=xxxx)

- Unable to execute refresh credentials function

  微服务日志STS SDK抛出如下异常，这是由于对端开启了STS的微服务认证，MapConnectCoreService&MapConnectCoCreateService未给该微服务配置ACL。

  解决方案：让对端给该微服务配置ACL。

  ```
  com.huawei.wisesecurity.sts.sdk.exception.StsException: Unable to execute refresh credentials function for service.identity=MapConnectCoreService&MapConnectCoCreateService
  ```

- STS对JDK/JRE有什么要求?

  建议使用最新的1.8版本JDK/JRE。

- invalid common name

  STS SDK抛出如下异常：

  ```
  InvocationException: code=403;msg=errorCode=1002;errorMessage=invalid common name: AppGalleryPromotionUserService
  ```

  这是由于CSE的yaml文件中配置的微服务名和使用的STS微服务证书中的微服务名不一致导致的，使用部署时采用STSAgent自动签发的证书，里面是标准的PBI服务名和微服务名。

- STS管理台敏感配置修改后如何刷新到微服务的配置文件中?

  敏感配置项是通过配置服务发布到微服务配置文件中。在STS管理台修改后，需要在部署服务上重新发布修改的配置项。

  - 在配置中心找到该配置项
  - 重新发布该配置项：

- fail to parser work key json file

  检查work_key.json格式是否正确，文件中的description，mac字段需要删除。

- Odd number of characters

  业务加载工作密钥文件的时候报Odd number of characters异常，这是由于在STS管理台配置的hex编码的密钥明文的少或多了一个char。Hex编码字符串必须是偶数个字符。

- fail to get master key from sts

  业务集成了STS 1.0的认证能力，需要从STS 1.0 Server上获取主密钥失败。STS 1.0 Server是部署在业务集群中，注册到业务同一个注册中心：

  - 在相应注册中心上检查WiseSecurity：SecurityTokenService微服务是否存在，如果不存在，则找到相应部署机器启动该微服务。
  - 检查业务的yaml文件中配置的environment参数和STS 1.0微服务中的environment是否一致，如果不一致则修改成一致。

- missed header x-sts-token / x-sts-session

  provider通过STS SDK认证请求时，从http消息中无法获取到相应的STS认证header信息。

  - 直接用postman访问，会缺少相应的认证凭据而失败。因此如果测试需要，可以将微服务的STS认证放通开关打开。放通开关配置在sts.properties配置文件中的sts.bypass.enable参数：true表示放通，false表示不放通，默认是不放通。

- consumer侧开启了放通，但provider侧没有放通。

- sts.properties文件中未配置sts.config.path指定STS微服务证书加载路径（如果是IDEA中执行的，检查一下target下是否存在sts.properties）。STS SDK默认加载sts/sts.properties，业务也可以通过jvm启动参数sts.properties修改该文件的路径。

  如果是通过NUWA初始化STS的，检查yaml文件中是否有配置nuwa.security.sts.configPath。

  ```
  nuwa:
    security:
      sts:
        serverDomain: 10.33.102.109:8080
        configPath: /opt/huawei/certs/{ServiceName}/{MicroServiceName}/{MicroServiceName}.ini
  ```

  如果通过Java Bean方式初始化，检查是否有设置sts.config.path配置项。

  ```
  Properties properties = new Properties();
   properties.setProperty("sts.server.domain", "10.33.102.162:8080");
   properties.setProperty("sts.config.path", "D:/Test/ConsumerService/ConsumerMicroService1/
  ConsumerMicroService1.ini");
   StsKeyApi.initWith(properties);
  ```

- STS管理台录入敏感配置项使用什么加密算法加密。

  敏感配置项采用STS给业务分配的KeK/ServiceKek（256位）使用AES/GCM加密。

- Unable to resolve non-exist credentials

  StsCseClientFilter中发送请求时报如下异常和错误，这是由于无法获取到访问该微服务凭据抛出异常。检查服务初始化STS SDK时是否有异常（查看 logback或log4j中配置的com.huawei.wisesecurity.sts包路径的日志文件）。

- invalid certificate

  回答：确认服务名和证书名是一致的，证书是重新下发的，再判断业务连接的STS server1.0上的证书是否有修改。

  注意：需要确定当前环境连接的STS服务器是正确的，可以确认性能环境所对应的注册中心，然后在这个注册中心的portal界面上查找并确认STS 1.0是哪台服务器注册上去的。

- get sts token --Timestamp out of range/Timestamp is more earlier or later

  这是由于STS开启防重放攻击（和STS服务器之间时间差小于5分钟），检查业务服务器的时间是否正确。

- configId can not be blank

  部署服务或者NUWARutime获取敏感配置项时，请求要获取的敏感配置项的列表中有值为空。

- 管理台上服务下无法选到相应的微服务。

  在安全管理台上配置敏感配置项、证书时，服务下只显示已经在相应环境下在STS上注册过的微服务。需要让站点的业务SRE在运维管理台上进行微服务注册。

- KeyStoreException: empty private key or certificate

  出现该提示，是因为主机上STS实例证书（位于/opt/huawei/certs/{服务}/{微服务}/下）并不完整，很可能是由于前一次部署过程中出错导致，请检查该证书是否有效，可使用以下命令对证书进行检查。

  ```
  /opt/huawei/apps/wisesecurity/stsagent/stsagent check -s {服务} -m {微服务}
  ```

  如果提示校验错误信息，请重新执行stsagent-install部署步骤进行证书的重新申请。

- Unable to find SecretKey version=0

  业务代码中使用STS密钥解密非STS加密的数据导致的错误。

- Micro service xxx&yyy not registered

  微服务没有在STS管理台进行注册。

- Fail to read service info from keystore

  STS SDK初始化是需要从STS的身份证书jks文件中读取服务和微服务信息，加载jks文件的时候异常了。一般是该jks文件有异常。

  SDK会根据service ini文件中的subject的CN项作为微服务名读取证书文件 {微服务名}.sts.p12

  ```
  [cert] subject = C=CN,O=Huawei,OU=Huawei CBG Cloud
  MicroService,CN=SecurityTokenServiceDemoSelfConsumer ip = 127.0.0.1
  ```

- invalid timestamp

  检查业务主机的时间和标准时间差异是否超过5分钟。

- timeout awaiting response headers

  sts-go-sdk内的 ResponseHeaderTimeout参数值设置过小，目前默认为100，在初始化时传入参数“sts.responseHeader.timeout”修改该值大小。

- not an SSL/TLS record

  client和server两边SSL不一致：client通过HTTP访问，但Server是提供的HTTPS的服务。

  如果是 tomcat + NUWA/CSE，tomcat最终对外提供是http还是https是由tomcat的配置文件决定的，NUWA/CSE的yaml文件中定义的只是注册到注册中心告诉对端提供的是 http还是https。

- Some config items not found

  iac provider在plan阶段会去校验敏感配置项是否存在于STS管理台，如果没有录入敏感配置项，或者敏感配置项的坐标不正确，则报此错，需要业务自己检查坐标是否正确。

- fail to get provider ** credentials

  consumer集成sts-go-sdk在获取provider端访问凭据的时候失败，需要在provider端配置ACL，管理台录入ACL后，需等十分钟才生效。

  启动时如果发现类似错误：

  either 'jasypt.encryptor.password', one of ['jasypt.encryptor.private-key-string', 'jasypt.encryptor.private-key-location'] for asymmetric encryption, or one of ['jasypt.encryptor.gcm-secret-key-string', 'jasypt.encryptor.gcm-secret-key-location', 'jasypt.encryptor.gcm-secret-key-password'] for AES/GCM encryption must be provided for Password-based or Asymmetric encryption

  那么可以按两个方向排查：

  - 检查启动类是否添加@EnableStsEncryptableProperties ，需要添加该注解，才能初始化StsStringEncryptor对象。
  - 如果已经添加，那可能是触发自动解密的地方距离启动太早，导致 spring context 初始化还未完成，不能获取到StsStringEncryptor对象。例如，Eureka应用的bootstrap.yml中如果包含敏感配置项，就会报这个错，需要将敏感配置项移到其他配置文件中，即可解决。

- Unable to refresh credentials for service SecurityTokenService&SecurityTokenMicroService

  该问题是STS初始化的时候，从kek.json或者keys.json获取本地密钥失败，导致无法从STS-Server获取凭据。

  解决方法：找到密钥的缓存地址，/${HOME}/.sts/服务名/微服务名/kek.json，删除kek.json和keys.json文件，重新启动即可。

3.2 Cloud Map SDK

3.2.1 概述

Cloud Map SDK服务软件开发工具包是对服务发现（Cloud Map）服务提供的REST API进行的封装，以简化用户的开发工作。

Cloud Map是注册发现中心，主要用于注册发现微服务、中间件/数据库、一方/二方/三方服务，单体应用无需对接Cloud Map（微服务不使用任何中间件/数据库，也不依赖任何其他服务/微服务），用户直接调用Cloud Map SDK提供的接口函数即可实现使用Cloud Map服务业务能力的目的。

3.2.2 使用 Cloud Map SDK（Spring Cloud 框架）

引入 Cloud Map SDK

1. 引入STS

   Cloud Map依赖STS认证能力，接入Cloud Map必须接入STS，具体请参考**引入STS SDK**。

2. 引入Cloud Map

   在pom.xml中添加Cloud Map SDK依赖。

   - 将${cloudmap-sdk-version}替换成实际所使用的Cloud Map SDK版本。
   - 如果将SDK放到外部maven仓中，则只需要添加nuwa-cloudmap-core依赖。
   - 如果采用本地依赖的方式引入SDK，即手动将本地下载的SDK jar包引入到工程的lib目录下，还需要添加间接依赖：nuwa-cloudmap-spring-boot-starter、spring-cloud-starter-cloudmap-discovery。

   ```
   <dependency>
     <groupId>com.huawei.wisecloud.nuwa</groupId>
     <artifactId>nuwa-cloudmap-core</artifactId>
     <version>${cloudmap-sdk-version}</version>
   </dependency>
   ```

配置 Cloud Map

在微服务的application.yaml配置文件中添加以下配置项：

```
nuwa:
  cloudmap:
    read: cloudmap                          #使用Cloud Map方式进行微服务间通信
    clusterName: clusterName-example        #微服务注册到Cloud Map的集群
    provider:
      cluster: clusterName-example          #提供服务的其他同样被注册到Cloud Map的微服务集群名
    serverAddr: http://10.34.32.243:80      #Cloud Map访问地址
    version: 1.0.0.100                      #微服务版本号
    namespaceName: cn_dev_default           #Cloud Map访问命名空间
```

初始化 Cloud Map

在启动类中增加@EnableDiscoveryClient注解，同时在启动类中完成将RestTemplate放到spring容器中，后续微服务间调用就使用注册到spring容器中的RestTemplate，代码如下：

```
package com.huawei.demo.servicea;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

import com.huawei.wisesecurity.sts.springboot.security.annotation.EnableStsAutoInitialization;

@SpringBootApplication
@EnableStsAutoInitialization(value = "application.properties")
@EnableDiscoveryClient
public class ServiceASpringbootApplication {
    public static void main(String[] args) {
        SpringApplication.run(ServiceASpringbootApplication.class, args);
    }

    @Bean
    @LoadBalanced
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

微服务间调用：

```
public OrderInfo findOrder(String orderId, HttpServletRequest request) {
        OrderInfo orderInfo = orderMapper.getOrderById(orderId);
        String url = demoServiceAUrl + "/user/" + orderInfo.getUserId();
        HttpHeaders headers = createHeaders(request);
        HttpEntity<String> entity = new HttpEntity<>(null, headers);
        ResponseEntity<UserInfo> responseEntity = restTemplate.exchange(url, HttpMethod.GET, entity,
UserInfo.class); ;
        if (responseEntity != null) {
            orderInfo.setUserInfo(responseEntity.getBody());
        }
        return orderInfo;
    }
```

3.2.3 使用 Cloud Map SDK（NUWA 框架）

引入 Cloud Map SDK

NUWA框架的nuwa-core模块已经包含STS的SDK，只需要在pom.xml中以provided方式引入Cloud Map的插件即可。

```
<dependency>
   <groupId>com.huawei.wisecloud.nuwa</groupId>
   <artifactId>nuwa-gpaas-cloudmap</artifactId>
   <version>${nuwa-version}</version>
   <scope>provided</scope>
</dependency>
```

同时要在classpath: nuwa.boot.properties文件中添加nuwa-gpaas-cloudmap模块，否则部署时不会加载Cloud Map。

```
nuwa.system.module.loadingList=...,nuwa-gpaas-cloudmap,…
```

初始化 Cloud Map

使用NUWA框架，只要增加对应配置，框架即会完成Cloud Map的初始化。

这些配置需要写到nuwa框架可以读到的文件里，一般是nuwa-xxx.properties，nuwa-xxx.yaml文件。因为Cloud Map的Client全局单例，所以可以直接通过NuwaMapClientFactory.getNuwaMapClient()获取client对象，也可以在Spring上下文中获取NuwaMapClient的Bean。

1. 增加STS配置。
```
nuwa.security.sts.enable=true
nuwa.security.sts.serverDomain=10.33.102.162:8080
nuwa.security.sts.configPath=/opt/huawei/certs/serviceName/microServiceName/microServiceName.ini
```
2. 增加Cloud Map配置。
```
nuwa.cloudmap.serverAddr=http://10.34.32.243:80
nuwa.cloudmap.namespaceName=cn_dev_default
```

3.2.4 常见问题

表 3-2 Cloud Map 常见问题

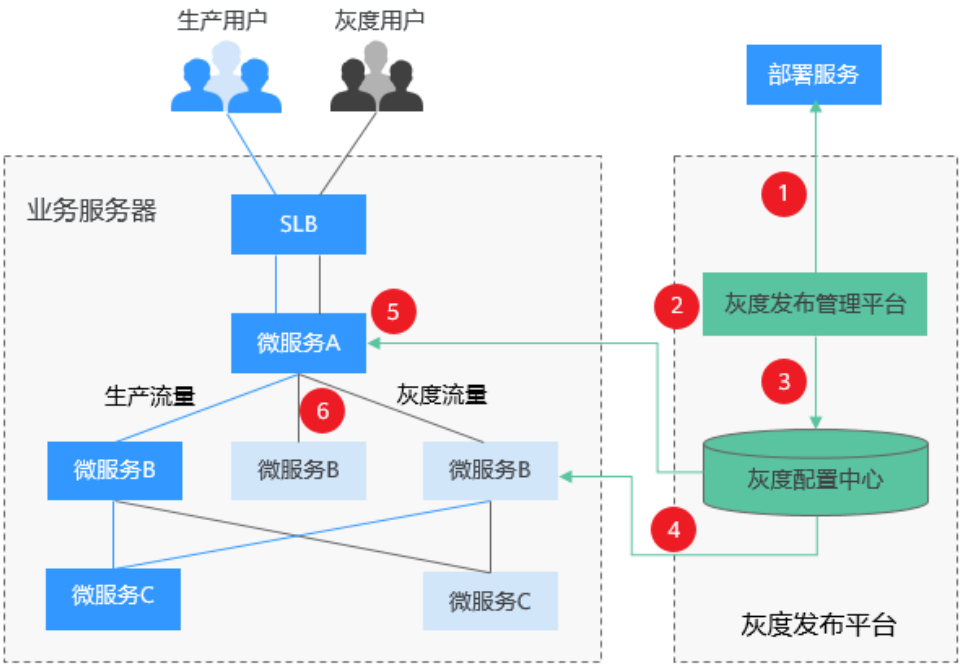
问题	报错原因	解决办法
“code”:403，no permission to access	业务没有权限访问中间件资源	1. 登录Cloud Map管理台查看资源的授权信息。 2. 如果自身的服务或微服务不在授权范围内，则没有访问权限，需要确认注册中间件资源时所设置的授权信息（一般是集群所属的服务才有权限）。 3. sds报错no permission，在sds管理台新增SDK配置时没有填写正确的微服务名。 Rainbow使用共享服务报错no permission，需要将Rainbow资源在sdk管理台注册到自己的服务下，而不是其他服务下。 微服务不为*，则只有一个微服务有权限。如果需要改成*，需要在相应管理台同步配置时不要选微服务名。
“code”:404，resource is not exist	业务要找的资源不存在	1. 先确认资源是否存在，登录Cloud Map管理台查找资源，如果不存在，参考开发指南注册资源。 2. 如果资源确实存在，请检查资源的属性和业务配置属性是否一致，如namespace、服务名、集群名等。 3. debug查看调用Cloud Map接口的数据（如serviceName）是否正常。 4. 共享服务的，请检查配置是否正确。 5. 检查NUWA版本是否过低，建议3.0.5.101以上。
Too many connections	低版本asynchttpclient连接数超过设置最大值（默认为cpu核数，不超过4）	1. 升级SDK版本号。 2. 修改asynchttpclient连接数。

3.3 Gray SDK

3.3.1 概述

系统架构

图 3-1 SLB 系统架构



系统架构说明：

1. 调用部署服务接口获取业务微服务列表，以及发布的微服务版本信息，用于配置不同微服务版本的灰度策略。
2. 运维在灰度发布管理台配置微服务灰度策略，支持配置同一个微服务不同版本的灰度规则，以及不同微服务相同的灰度规则。
3. 微服务的灰度策略同步到灰度配置中心。
4. 微服务集成灰度SDK，订阅微服务的灰度配置，保存到缓存和本地文件。
5. 灰度SDK执行灰度策略 ，根据微服务传入的灰度参数对象，SDK执行灰度规则，返回匹配结果以及规则对应的微服务版本号。
6. 微服务根据灰度SDK执行结果进行灰度路由，消费端根据灰度SDK返回结果调用对应的微服务B（ 生产、灰度版本1 、灰度版本2 ）。

3.3.2 使用 Gray SDK

本文介绍如何在NUWA框架下使用Gray SDK，Spring Cloud框架下暂时不支持使用Gray SDK。

使用Gray SDK时，需要先在AppStage运维中心SLB服务中配置好灰度规则，再启动Gray SDK，否则灰度规则不生效。

引入 Gray SDK

通过NUWA的nuwa-apaas-graysdk间接集成，集成该SDK之后，可以使用Gray SDK。

在pom.xml中添加STS依赖。

将${nuwa-sdk-version}替换成实际所使用的Gray SDK版本。

```
<dependency>
    <groupId>com.huawei.wisecloud.nuwa</groupId>
    <artifactId>nuwa-apaas-graysdk</artifactId>
    <version>${nuwa-sdk-version}</version>
    <scope>system</scope>
    <systemPath>${project.basedir}/lib/nuwa-apaas-graysdk-3.1.5.100.jar</systemPath>
</dependency>
```

初始化 SLB

- 业务自行解析好配置，作为Properties入参传入。

  GraySDKManager.getInstance().init(Properties params, Object lbsServiceFactory)。

  表 3-3 参数说明

参数	说明	传空说明
Properties params	将Properties对象传入	将不能使用Gray SDK功能。 不能传空。
Object lbsServiceFactory	lbsServiceFactory对象传入	不能使用LBS国家省市IP库功能。 在业务不使用LBS国家省市灰度时，可传空。

- 业务指定好Properties路径，properties文件路径传入。

  GraySDKManager.getInstance().initWithPropertiesPath (String path, Object lbsServiceFactory)。

表 3-4 参数说明

参数	说明	传空说明
String path	将Properties文件全路径传入。	传空时，会按照以下顺序尝试补参： 1. 尝试以系统参数指定的路径找文件System.getProperty(“graysdk.properties.file_path”)。 2. 尝试从classpath的文件系统路径下找graysdk.properties或者graysdk-config.properties。
Object lbsServiceFactory	LbsServiceFactory对象传入。	不能使用LBS国家省市ip库功能。 在业务不使用LBS国家省市灰度时，可传空。

开启灰度过滤器

在消费端的微服务上配置好properties文件中以下开关：

```
# enable CseGrayServerListFilter or not
wgp.enableCseGrayFilter=true
```

服务消费端集成示例

照常通过CSE调用服务提供者，无需额外编码。

例如，参考CSE的消费端开发，从消费端调用服务提供端（ demoB ），无需额外修改。

```
package com.service.demo.controller;

import org.springframework.stereotype.Component;
import org.apache.servicecomb.provider.springmvc.reference.RestTemplateBuilder;
import org.springframework.web.client.RestTemplate;
@Component
publicclass DemoDelegate {

public String helloworld(String name){

if ( null == name || "".equals(name))
{
return "thename is empty, no need send to beckend MS";
}

RestTemplate restTemplate = RestTemplateBuilder.create();
String rslt = "";
String qryStr = "";
qryStr = "name=" + name;

// to call another MicroService, name is demoB. path:/demo,interface:helloworld
rslt = restTemplate
```



```
.getForObject("cse://demoB/demo/helloworld?" + qryStr, String.class);
return rslt;
}
}
```

3.3.3 常见问题

常见问题

- SDK初始化失败

  SDK初始化失败，graysdk的run和debug日志会打印初始化失败原因，一般原因有如下几种：
  - ETCD服务器配置错误（检查conf/graysdk.properties文件中server配置是否正确）。
  - ETCD未启动或者网络原因不可用（检查网络以及ETCD是否正常）。
  - ETCD中不存在此微服务名称的灰度配置。
  - ETCD未启用用户名/密码认证方式，但是SDK配置了认证（检查conf/graysdk.properties文件是否配置了username）。
  - jsonRuleFilePath没有配置。
- 灰度规则匹配失败

  调用SDK接口执行灰度规则匹配失败，一般原因有如下几种：
  - 实际传入的参数和值，不满足配置的微服务灰度规则。
  - 灰度开关关闭（检查conf/graysdk.properties文件graySwitch是否为1）。
  - 灰度规则执行异常（检查graysdk的debug日志）。
  - 联系灰度平台开发人员定位。
- 灰度参数的设置

  消费端需要在CSE的invocation设置想要进行灰度的参数，有两种方法：
  - 确定需要进行灰度的参数 ，在后端的接口中使用@RequestParam参数来指定。
  - 业务直接在消费端设置参数，例如：ContextUtils.getInvocationContext().addContext("x-is-gray", "1")。

  第一种方式直接在请求流量的URL中通过设置参数的值来标识灰度流量，第二种是业务在调用SDK之前改变invocation中参数的值来标识灰度。
- 后端灰度节点的判断

  后端的节点在自己的microservice.yaml文件中设置参数描述（参照**CSE开发网站**）。
  - 按照版本筛选灰度节点，需要在yaml中的service_description中先定义版本号，然后在管理台上选择按照版本进行灰度，填入定义的版本。
  - 按照自定义参数筛选灰度节点，需要在yaml中的instance_description中先自定义灰度参数，并填入值，接着在管理台选择按照自定义参数进行灰度，填入自定义的参数。

附录

- SDK错误码

  在调用SDK接口时，出现异常情况时SDK会返回一个GrayException异常，可以通过e.toString()查看详细的异常说明，通过e.getCode()获取异常错误码，同时通过SDK的debug日志可以看到更详细的异常错误信息。

表 3-5 SDK 错误码

异常编码	异常原因	异常处理方法
13300	内部处理错误，一般为配置错误导致。	查看graysdk的debug日志，查看error日志，同时联系灰度平台开发人员定位。
13301	参数错误，一般为传入SDK的微服务名称为空或者空串。	1. 检查graysdk.properties文件etcd.serviceName配置是否正确。 2. 检查graysdk的run日志，查看sdk是否初始化成功。
13302	cloudsoa security解密密码失败。	检查graysdk.properties文件etcd.password是否配置正确。
13303	操作ETCD失败（读取ETCD配置触发异常）。	1. 检查graysdk.properties文件etcd.server配置是否正确。 2. 检查ETCD服务器是否工作正常。

- 性能统计日志

  日志字段默认采用“|” 分隔，一条日志一行，采用“\n”结尾，默认1s（可配置）打印一条。

表 3-6 节点级日志

字段名称	必选（M）/可选（O）	字段说明
时间	M	YYYY-MM-DD<1SP>hh:mm:ss[.SSS]
请求次数	M	周期值，每个周期清零，重新计数。
未进行匹配次数	M	未进行灰度匹配的请求次数。一般为灰度完成、灰度暂停、未开启灰度等。 周期值，每个周期清零，重新计数。
匹配成功次数	M	周期值，每个周期清零，重新计数。
匹配失败次数	M	周期值，每个周期清零，重新计数。
匹配异常次数	M	周期值，每个周期清零，重新计数。

字段名称	必选（M）/可选（O）	字段说明
匹配超时次数	M	周期值，每个周期清零，重新计数。
请求平均时延	M	周期内请求的平均时延，单位：微秒。
请求最大时延	M	周期内请求的最大时延，单位：微秒。
请求最小时延	M	周期内请求的最小时延，单位：微秒。

表 3-7 服务级日志

字段名称	必选（M）/可选（O）	字段说明
时间	M	YYYY-MM-DD<1SP>hh:mm:ss[.SSS]
微服务名	M	被调用的微服务名称。
灰度服务名	M	为被调用的微服务名称，或为被调用的微服务的映射服务名称。
请求次数	M	周期值，每个周期清零，重新计数。
未进行匹配次数	M	未进行灰度匹配的请求次数。一般为灰度完成、灰度暂停、未开启灰度等。 周期值，每个周期清零，重新计数。
匹配成功次数	M	周期值，每个周期清零，重新计数。
匹配失败次数	M	周期值，每个周期清零，重新计数。
匹配异常次数	M	周期值，每个周期清零，重新计数。
匹配超时次数	M	周期值，每个周期清零，重新计数。
请求平均时延	M	周期内请求的平均时延，单位：微秒。

字段名称	必选（M）/可选（O）	字段说明
请求最大时延	M	周期内请求的最大时延，单位：微秒。
请求最小时延	M	周期内请求的最小时延，单位：微秒。

- 告警日志
  日志字段默认采用“|” 分隔，一条日志一行，采用“\n”结尾。

字段名称	必选（M）/可选（O）	字段说明
时间	M	YYYY-MM-DD<1SP>hh:mm:ss[.SSS]
告警ID	M	graysdk000000001-4
告警名称	M	graysdk000000001：gray servers all down alarm graysdk000000002：etcd connect error graysdk000000003：gray rule execute exception graysdk000000004：gray rule execute timeout
本机IP	M	本机IP
告警类别	M	Firing：告警发生 Resolved：告警解除
详细信息	M	告警详细信息
告警方式	M	自动告警
开始时间	M	Firing时，为告警发生时间，Resolved时为0。
结束时间	M	Resolved时，为告警解除时间，Firing时为0。

service.json文件

```
{
  "enableGrayMapping": "1",
  "services": [{
     "name": "demoA",
     "wgpServiceName": "demoB"
  },
 {
    "name": "demoC",
```



```
    "wgpServiceName": "demoB" }]
}
```

NUWA配置项

```
nuwa:
  wgp:
    graySwitch: 1
    jsonRuleFilePath: /opt/huawei/data/
    serviceName: xxxxxx(业务自己配置)
    sdkServiceUrl: xxxxxxx
    enableCseGrayFilter: true
    enableCommonGrayFilter: false
    reportUrl: http://10.28.0.181:18080/dispatchProxy/v1
```

3.3.4 日志处理

介绍SDK提供的日志配置方法以及日志格式。

日志级别配置

- 日志组件

  SDK以SLF接口输出日志，默认使用logback输出日志，引入了logback-core-***.jar和logback-classic-***.jar这两个jar包，SLF会绑定到logback输出日志。

- 日志配置示例

  这里以logback日志组件配置为例，相关配置参考如下，如果业务用的log4j2，也可以参考这里对应的logger和appender来配置。

  – 运行日志、调试日志

```
    <appender name="wgp_debug_log"
        class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>../logs/graysdk/debug/debug.log</file>
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss:SSS}|%thread|%level|%logger{0}|:%msg%n</
pattern>
        </encoder>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>../logs/graysdk/debug/debug-%d{yyyyMMddHH}.log
            </fileNamePattern>
            <maxHistory>24</maxHistory>
        </rollingPolicy>
        <timeBasedFileNamingAndTriggeringPolicy
            class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
            <maxFileSize>100MB</maxFileSize>
        </timeBasedFileNamingAndTriggeringPolicy>
    </appender>
    <appender name="wgp_run_log"
        class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>../logs/graysdk/run/run.log</file>
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss:SSS}|%level|%msg%n</pattern>
        </encoder>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>../logs/graysdk/run/run-%d{yyyyMMddHH}.log
            </fileNamePattern>
            <maxHistory>24</maxHistory>
        </rollingPolicy>
        <timeBasedFileNamingAndTriggeringPolicy
            class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
            <maxFileSize>100MB</maxFileSize>
        </timeBasedFileNamingAndTriggeringPolicy>
    </appender>
    <logger name="com.huawei" additivity="false">
        <level value="INFO" />
        <appender-ref ref="debug_log" />
    </logger>
```

- 性能统计日志

  logback.xml中定义性能统计日志

```
<appender name="wgp_stat_log" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>../logs/graysdk/stat/stat.log</file>
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss:SSS}|%level|%msg%n</pattern>
        </encoder>
        <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
            <fileNamePattern>../logs/graysdk/stat/stat-%d{yyyy-MM-dd}.%i.log.zip</
fileNamePattern>
            <maxFileSize>100MB</maxFileSize>
            <maxHistory>7</maxHistory>          <!-- log save days -->
            <totalSizeCap>10GB</totalSizeCap>   <!-- total log space size -->
        </rollingPolicy>
</appender>
<appender name="wgp_stat_service_log"
class="ch.qos.logback.core.rolling.RollingFileAppender">
<file>../graysdk/stat/stat_service.log</file>
<encoder>
<pattern>%d{yyyy-MM-dd HH:mm:ss:SSS}|%msg%n</pattern>
</encoder>
<rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
<fileNamePattern>../graysdk/stat/stat_service-%d{yyyy-MM-dd}.%i.log.zip</fileNamePattern>
<maxFileSize>100MB</maxFileSize>
<maxHistory>7</maxHistory>
<totalSizeCap>10GB</totalSizeCap>
</rollingPolicy>
</appender>
    <logger name="WgpStatLogger" additivity="false">
        <level value="all" />
        <appender-ref ref="stat_log" />
</logger>
<logger name="WgpStatServiceLogger" additivity="false">
<level value="all" />
<appender-ref ref="wgp_stat_service_log" />
</logger>
```

- 告警日志

  logback.xml中定义告警日志

```
<appender name="wgp_alarm_log" class="ch.qos.logback.core.rolling.RollingFileAppender">
<file>../logs/graysdk/alarm/alarm.log</file>
<encoder>
    <pattern>%d{yyyy-MM-dd HH:mm:ss:SSS}|%msg%n</pattern>
</encoder>
<rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
    <fileNamePattern>../logs/graysdk/alarm/alarm-%d{yyyy-MM-dd}.%i.log.zip</
fileNamePattern>
    <maxFileSize>100MB</maxFileSize>
    <maxHistory>7</maxHistory>          <!-- log save days -->
    <totalSizeCap>10GB</totalSizeCap>   <!-- total log space size -->
</rollingPolicy>
</appender>
<logger name="WgpAlarmLogger" additivity="false">
<level value="all" />
<appender-ref ref="stat_log" />
</logger>
```

- 日志说明

  SDK默认会产生debug日志和run日志。

- SDK初始化成功

  SDK初始化成功，表示已经从ETCD获取到了指定微服务名称的灰度配置，同时会在conf目录产生{serviceName}_grayrule.json的本地灰度规则备份文件。

3.4 Rainbow SDK

3.4.1 概述

Rainbow SDK构建云原生DevOps全流程可信build-in的数据库治理解决方案 。

- 提供MySQL/Cassandra/GaussDB/DRDS全流程设计、开发、发布、运维（管理、治理、诊断）方案。
- 可信build-in：过程可信，结果可信，接入安全（无人工接入密码），操作（资源高危操作）安全。

3.4.2 使用 Rainbow SDK（Spring Cloud 框架）

引入 Rainbow SDK

1. Rainbow SDK依赖Cloud Map的注册和发现能力，在引入Rainbow SDK之前，要先引入STS SDK和Cloud Map SDK，并完成STS和Cloud Map的初始化，具体请参见**使用STS SDK（Spring Cloud框架）**和**使用Cloud Map SDK（Spring Cloud框架）**。

   使用Cloud Map进行数据库注册和发现时，需要先在WiseDBA管理台配置连接信息，并注册到Cloud Map中，Rainbow SDK连接数据库的过程中，直接调用Cloud Map中的注册信息，具体操作请参见**新增数据库SDK配置并注册到Cloud Map**。
2. 引入Rainbow SDK。

   在pom.xml中添加Rainbow SDK依赖。
   - 将${rainbow-sdk-version}替换成实际所使用的Rainbow SDK版本。
   - 如果将SDK放到外部maven仓中，则只需要添加rainbow-proxy依赖。
   - 如果采用本地依赖的方式引入SDK，即手动将本地下载的SDK jar包引入到工程的lib目录下，还需要添加间接依赖：rainbow-core-drds、gpaas-middleware-common。

```
<dependency>
   <groupId>com.huawei.wisecloud.nuwa</groupId>
   <artifactId>rainbow-proxy</artifactId>
   <version>${rainbow-sdk-version}</version>
</dependency>
```

初始化 Rainbow SDK

代码如下：

```
package com.huawei.demo.serviceb.config;

import javax.sql.DataSource;

import org.mybatis.spring.SqlSessionFactoryBean;
import org.mybatis.spring.mapper.MapperScannerConfigurer;
import org.springframework.context.EnvironmentAware;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.DependsOn;
import org.springframework.context.annotation.Import;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.core.env.Environment;
```

```
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.datasource.DataSourceTransactionManager;
import org.springframework.transaction.support.TransactionTemplate;

import com.huawei.nuwa.map.client.NuwaMapClient;
import com.huawei.nuwa.map.spring.boot.autoconfigure.NuwaCloudMapAutoConfiguration;
import com.huawei.rainbow.jdbc.DbGroupDataSource;
import com.huawei.wisesecurity.sts.springboot.security.configuration.StsEncryptablePropertiesConfiguration;

/**
 * Rainbow启动
 */
@Configuration
@Order(Ordered.HIGHEST_PRECEDENCE)
@Import(value = {NuwaCloudMapAutoConfiguration.class, StsEncryptablePropertiesConfiguration.class})
public class ApplicationAutoConfig implements EnvironmentAware {
    private Environment environment;

    /**
     * Rainbow需要依赖STS和Cloud Map启动，创建dataSource
     *
     * @param client NuwaMapClient值对象
     * @return DataSource 初始化完成的数据源数据源
     */
    @Bean
    @DependsOn({"stsBootStrap"})
    public DataSource createDataSource(NuwaMapClient client) {
        DbGroupDataSource dataSource = new DbGroupDataSource();
        dataSource.setUseSts(true);
        dataSource.setAppName(environment.getProperty("wiseDba.appName"));
        dataSource.setDbGroupKey(environment.getProperty("wiseDba.groupKey"));
        dataSource.setDbName(environment.getProperty("wiseDba.dbName"));
        dataSource.init();
        return dataSource;
    }

    /**
     * 事务管理创建
     *
     * @param dataSource 数据源
     * @return DataSourceTransactionManager 事务管理
     */
    @Bean
    public DataSourceTransactionManager createTransaction(DataSource dataSource) {
        DataSourceTransactionManager transaction = new DataSourceTransactionManager();
        transaction.setDataSource(dataSource);
        return transaction;
    }

    @Bean
    public JdbcTemplate createJdbcTemplate(DataSource dataSource) {
        return new JdbcTemplate(dataSource);
    }

    @Bean("sqlSessionFactoryBean")
    public SqlSessionFactoryBean createMybatisSqlSessionFactoryBean(DataSource dataSource) {
        SqlSessionFactoryBean sessionFactoryBean = new SqlSessionFactoryBean();
        sessionFactoryBean.setDataSource(dataSource);

        // 数据源配置项
        org.apache.ibatis.session.Configuration configuration = new org.apache.ibatis.session.Configuration();

        // 允许jdbc自动生成主键
        configuration.setUseGeneratedKeys(true);

        // 使用列标签代替列名
        configuration.setUseColumnLabel(true);

        // 打开下划线命名自动转换为驼峰命名开关
```



```
        configuration.setMapUnderscoreToCamelCase(true);
        sessionFactoryBean.setConfiguration(configuration);
        return sessionFactoryBean;
    }

    @Bean
    public TransactionTemplate createTransactionTemplate(DataSourceTransactionManager
dataSourceTransactionManager) {
        TransactionTemplate template = new TransactionTemplate();
        template.setTransactionManager(dataSourceTransactionManager);
        return template;
    }

    @Bean
    public MapperScannerConfigurer createMapperScannerConfigurer() {
        MapperScannerConfigurer configurer = new MapperScannerConfigurer();
        configurer.setBasePackage("com.huawei.demo");
        configurer.setSqlSessionFactoryBeanName("sqlSessionFactoryBean");
        return configurer;
    }

    @Override
    public void setEnvironment(Environment environment) {
        this.environment = environment;
```

3.4.3 使用 Rainbow SDK（NUWA 框架）

引入 Rainbow SDK

1. Rainbow SDK依赖Cloud Map的注册和发现能力，在引入Rainbow SDK之前，要先引入STS SDK和Cloud Map SDK，并完成STS和Cloud Map的初始化，具体请参见**使用STS SDK（NUWA框架）**和**使用Cloud Map SDK（NUWA框架）**。

   使用Cloud Map进行数据库注册和发现时，需要先在WiseDBA管理台配置连接信息，并注册到Cloud Map中，Rainbow SDK连接数据库的过程中，直接调用Cloud Map中的注册信息，具体操作请参见**新增数据库SDK配置并注册到Cloud Map**。

2. 引入Rainbow SDK

   在pom.xml中添加Rainbow SDK依赖。

   ```
   <dependency>
       <groupId>com.huawei.wisecloud.nuwa</groupId>
       <artifactId>nuwa-gpaas-rainbowproxy </artifactId>
       <version>${nuwa-version}</version>
       <scope>provided</scope>
   </dependency>
   ```

配置 Rainbow

修改Rainbow SDK配置，切换SDK数据源，新增Cloud Map配置、STS配置，配置如下：

```
nuwa:
  sts:
    serverDomain: 10.33.102.162:8080
    configPath: certs/WiseCloudNuwaService/WiseCloudNuwaCloudMapAdminService/
WiseCloudNuwaCloudMapAdminService.ini
    enable: true
  cloudmap:
    serverAddr: http://10.33.113.125:8080
    namespaceName: cn_dev_default
  rainbow:
    db0:
      datasource:
        beanName: rainbowDs
        dataSourceName: xxxx      //与运维中心WiseDBA服务SDK配置界面的值保持一致
```

```
    appName: xxx              //业务的服务名
    dbGroupKey: xxx           //实例名称
    dbName: xxxxx             //Schema名称
    connectionProperties:
characterEncoding=utf8;connectTimeout=10000;socketTimeout=10000;autoReconnect=true;useUnicode=true;
serverTimezone=Asia/Shanghai
    useSts: true
  mybatis:
    base-package: com.huawei.nuwa.map.demo.consumer.middleware.dao
    mapper-locations: classpath:mapper/rainbow/*.xml
```

3.4.4 常见问题

异常 1：Username is Emty!

异常：ERROR com.huawei.rainbow.utils.parser.DbNodeConfParser:46 - [] - appConfStr is not json format

原因：业务直接往ETCD里面set值，但set格式不是有效的json格式。

解决：使用工具检查，是否使用正确的Json串。

异常 2：Access denied for user

异常：java.sql.SQLException: Access denied for user 'xxxx'@'xx.xx.xx.xx' (using password: YES)

- 原因1：密码错误或者服务器分配权限错误。现网问题定位占了60%左右。数据库登录密码是否使用了明文！

  定位：

  a. 请认真检查密码配置，检查MySQL权限是否分配正确。

  b. 通过dump内存，搜索相关关键字判断解密字符为需要的字符串。

  解决：使用正确的密码。

- 原因2：或者是业务账号配错。

  解决：删除SDK配置信息，重新在wisedba新建业务账号，在SDK配置重新录入配置信息。

异常 3：NodeDataSource can't init 或者 init dataSource Params Error!

异常：com.huawei.rainbow.exception.RainbowRunTimeException: NodeDataSource can't init: dsKey=xx.xx.xx.xx:3306@mysql或者Caused by: com.huawei.rainbow.exception.DbNodeInitialException: [ConfigError]init dataSource Params Error! config is

- 原因1：大部分就是密码为null或者空串。

  定位：

  a. 请认真检查密码配置是否解密成功。通过搜索Rainbow日志关键字：Password is Empty! 请检查解密类是否继承了基类com.huawei.rainbow.utils.PasswordCoder。

  b. 搜索rainbow日志关键字：DbNodeConfig Check，查看是否校验失败。

  解决：使用正确的密码。

- 原因2：如果用的是cloud+stsl连的高斯数据库，原因是没有STS的调用步骤。
  通过搜索rainbow日志关键字：Password is Empty!
  解决：正确调用STS。
- 原因3：本地连接报这个错。
  解决：
  a. 先把配置信息删除，给Schema创建新的业务账号，新增配置信息。
  b. 把config目录下的rainbow.properties里面的ETCD地址和密码注释掉。

异常 4：dbGroupKey can not be null

异常：ParameterErrorException： dbGroupKey can not be null, length need bigger than 0（或者类似参数异常错误）。

原因1：dbGroupKey为null或者空串。 必填字段会做参数校验，请检查相关参数是否设置。

原因2：确定是否在rainbow.properties 中配置ETCD地址。

解决： 找到设置参数的地方，咨询核对配置的appName，dbGroupKey，dbName是否正确。

异常 5：Key not found, cause 或者 item lose

异常：类似 Key not found, cause: /Dbmonitor/status/DB/Services/xzjDB/xzjDB/activeDBs/mysql222, at index: 211562166

或者： item lose 关键字

或者：Caused by: com.huawei.rainbow.exception.ParameterErrorException: xxxx can not be null, length need bigger than 0

- 原因1：检查ETCD的IP配置。
- 原因2：在DCG导入的监控：是否监控成功（管理台亮绿灯），如果没有成功，一般是dbmonitor的权限没有配置正确，请检查。如果已经成功，查看前台的权重是否生成？没有生成则需要到后台查看。
  解决：查看DCG连接池配置信息是否生成，若无连接池配置信息，则手动添加。
- 原因3： 配置的appName(DCG界面的serviceName)，appGroupKey(DCG的ClusterName)，DBName(DCG的dbNname)是否配置正确。经常有业务将appName、appGroupKey与DCG配置的值搞得不一致，从而启动不了。
- 原因4：rainbow.properties文件路径没有读到。 关键字 "no etcd IP configured" 。
- 原因5：业务检查下，是否存在三方件引入的netty版本不一致导致冲突，检查netty-all版本和单独依赖的netty-handle等版本是否一致，如果不一致请业务统一升级到最新版本保持一致。
- 原因6：业务集成Rainbow使用的是oneJar的方式，在oneJar内和外面都定义了rainbow.properties 文件，存在冲突问题。验证方式：logback配置 jetcd日志com.huawei.dcs.jetcd、com.huawei.wisecloud.jetcd为INFO级别连接时会打印ETCD的地址，如：

```
<AsyncLogger name="com.huawei.dcs.jetcd" level="INFO" includeLocation="true" additivity="false">
    <AppenderRef ref="rainbow-common"/>
</AsyncLogger>
<AsyncLogger name="com.huawei.wisecloud.jetcd" level="INFO" includeLocation="true"
```



```
additivity="false">
    <AppenderRef ref="rainbow-common"/>
</AsyncLogger>
```

定位：

1. 通过命令检查ETCD里面是否已经存在相关值。

   curl http://{etcdip:port}/v2/keys/Dbmonitor/conf/DB/Services/{appName}/{dbGroupKey}/DBnames/{dbName}/{IP:port}

   curl http://{etcdip:port}/v2/keys/Dbmonitor/status/DB/Services/{appName}/{dbGroupKey}/activeDBs/{dbName}

   {xxx}为变量，替换成实际的字段。

2. rainbow.properties文件必须放到classpath目录下，否则读取不到。classpath目录下不能有多级目录，如classpath:db/目录。

解决：检查配置的 appName，appGroupKey，DBName字段是否正确。配置的ETCD地址是否需要连接的地址。appName对应DCG的ServiceName，appGroupKey对应instanceName。业务配置的appName是老的字段，导致报此错误。

异常 6：datasource.properties 没有更新，或者没有拉取到某个数据库配置

原因1：ETCD地址是否配错。

原因2：检查是否配置了DBDataSource的懒加载模式，只有当用数据库的时候才初始化，拉取配置。

定位：

1. 检查ETCD地址是否配错，查看日志是否有ETCD连接相关的错误。如果有，请修改重启。
2. 检查是否配置了DBDataSource的懒加载模式、只有当用数据库的时候才初始化，拉取配置。如果是，触发一次数据库操作，看下是否更新。

异常 7：启动时报 ETCD fault，can't not connect

原因1：ETCD连接不上。

原因2：调用close（ ）方法后，再调用了get etcd的方法。

定位：

1. 检查rainbow.properties文件路径有没有读到。 关键字 "no etcd IP configured" 。

2. 检查日志是否打印“close rainbow instance”关闭了Rainbow。

异常 9：启动时报 java.sql.SQLException: NodeDataSource XXXX@XXXX has closed

原因：Rainbow已经调用了close方法，导致无数据源。检查nuwa.out是不是有其他失败异常。

定位：检查日志中是否打印了“close rainbow instance”关闭了Rainbow。初始化失败时也会调用close方法。

异常 10：启动时报 Could not create connection to database server. Attempted reconnect 3 times. Giving up

原因：初始化失败，ping一下检查是否数据库是否联通。

3.4.5 日志处理

logger 名称和描述

Rainbow SDK需要配置的logger名称和描述如表3-8所示。

表 3-8 logger 说明

Logger名称	建议日志级别	说明
com.huawei.rainbow com.huawei.wisecloud.jetcd	INFO	Rainbow SDK运行日志，建议打开，方便定位问题。
com.huawei.dcg.rainbow.switch	INFO	Rainbow双云状态大数据日志。
com.huawei.dcg.rainbow.access	INFO	Rainbow SDK接入大数据日志。

日志文件配置样例

- logback.xml

  主要配置appender和logger，业务可以根据需要配置。

  file属性需要根据业务的路径进行配置。

```
<appender name="RAINBOW_LOG" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>../log/rainbow/rainbow.log</file>  <!-- 路径自己配置-->
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS}| [%thread]|%-5level|[%file:%line]|%p|%msg%n</
pattern>
        </encoder>
        <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
            <minIndex>1</minIndex>
            <maxIndex>10</maxIndex>
            <FileNamePattern>../log/rainbow/rainbow.log.%i</FileNamePattern>
        </rollingPolicy>
        <triggeringPolicy  class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
            <maxFileSize>20MB</maxFileSize>
        </triggeringPolicy>
</appender>

<appender name="RAIBOW_SWITCH_LOG" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>../log/rainbow/rainbow_switch.log</file>
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS}|%p|%msg%n</pattern>
        </encoder>
        <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
            <minIndex>1</minIndex>
            <maxIndex>10</maxIndex>
            <FileNamePattern>../log/rainbow/rainbow_switch.log.%i</FileNamePattern>
        </rollingPolicy>
        <triggeringPolicy  class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
```

```
            <maxFileSize>20MB</maxFileSize>
        </triggeringPolicy>
</appender>

<appender name="RAIBOW_STAT_LOG" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>../log/rainbow/rainbow_stat.log</file>
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS}|%p|%msg%n</pattern>
        </encoder>
        <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
            <minIndex>1</minIndex>
            <maxIndex>10</maxIndex>
            <FileNamePattern>../log/rainbow/rainbow_stat.log.%i</FileNamePattern>
        </rollingPolicy>
        <triggeringPolicy  class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
            <maxFileSize>20MB</maxFileSize>
        </triggeringPolicy>
</appender>

<logger name="com.huawei.rainbow" level="INFO" additivity="false">
                <appender-ref ref="RAINBOW_LOG" />
</logger>
<logger name="com.huawei.wisecloud.jetcd" level="INFO" additivity="false">
                <appender-ref ref="RAINBOW_LOG" />
</logger>
<logger name="com.huawei.dcg.rainbow.switch" level="INFO" additivity="false">
                <appender-ref ref="RAIBOW_SWITCH_LOG" />
</logger>
<logger name="com.huawei.rainbow.stat" level="INFO" additivity="false">
                <appender-ref ref="RAIBOW_STAT_LOG" />
</logger>
```

- log4j2.xml

  主要配置appender和logger：

  fileName属性需要根据业务的路径进行配置。

```
<RollingRandomAccessFile name="rainbow_log"
          fileName="${logDir}/rainbow/rainbow.log"
          filePattern="${logDir}/rainbow/rainbow.log.%d{yyyy-MM-dd}">
          <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS}|[%t]|%-5level|[%F:%L]|%p|%msg
%n"/>
          <Policies>
             <TimeBasedTriggeringPolicy/>
          </Policies>
</RollingRandomAccessFile>

<RollingRandomAccessFile name="rainbow_stat"
          fileName="${logDir}/rainbow/rainbow_stat.log"
          filePattern="${logDir}/rainbow/rainbow_stat.log.%d{yyyy-MM-dd}">
          <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS}|%msg%n"/>
          <Policies>
             <TimeBasedTriggeringPolicy/>
          </Policies>
</RollingRandomAccessFile>

<RollingRandomAccessFile name="rainbow_switch"
          fileName="${logDir}/rainbow/rainbow_switch.log"
          filePattern="${logDir}/rainbow/rainbow_switch.log.%d{yyyy-MM-dd}">
          <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS}|%msg%n"/>
          <Policies>
             <TimeBasedTriggeringPolicy/>
          </Policies>
</RollingRandomAccessFile>

<Logger name="com.huawei.rainbow" level="INFO"  additivity="false"
         includeLocation="true">
         <AppenderRef ref="rainbow_log" />
</Logger>
<Logger name="com.huawei.wisecloud.jetcd" level="INFO"  additivity="false"
```



```
        includeLocation="true">
        <AppenderRef ref="rainbow_log" />
</Logger>
<Logger name="com.huawei.dcg.rainbow.switch" level="INFO"  additivity="false"
        includeLocation="true">
        <AppenderRef ref="rainbow_switch" />
</Logger>
<Logger name="com.huawei.rainbow.stat.RainbowStatFilter" level="info"  additivity="false"
        includeLocation="true">
        <AppenderRef ref="rainbow_stat" />
</Logger>
```

4 AI 原生应用引擎 SDK API

4.1 版本变更记录

表 4-1 版本变更记录

版本	变更类型	说明
0.0.1	-	第一次发布。

4.2 SDK 概述

AI原生应用引擎SDK面向开发者提供了一套搭建原生应用的Python SDK，包含了模型调用，知识获取，工具调用等功能。开发者可以使用SDK调用AI原生应用引擎的各种能力，帮助开发者快速构建大模型应用。

表 4-2 模块说明

序号	模块	功能
1	HttpxClient	负责发送HTTP请求的基类。
2	ModelRouter	负责模型调用，包括文本对话、文本向量化等模型。
3	PromptTemplate	负责提示语模板的构建和初始化等。
4	KnowledgeRetriever	提供了知识库检索的能力，可以检索开发者的知识库，提取相关的信息。
5	ApplicationCenter	提供了开发者调用部署在平台的应用的能力。

4.3 快速开始

下载 AI 原生应用引擎 SDK

AI原生应用引擎SDK获取地址：**wiseagent-dev-sdk-python**。您也可以参考**下载SDK**获取AI原生应用引擎SDK并进行完整性校验。

AI原生应用引擎SDK获取方法请参考。

安装 AI 原生应用引擎 SDK

若已经安装过最新版本SDK，可以忽略此步骤。

执行如下命令，快速安装。

- 将${version}替换成实际所使用的SDK版本。
- 如果采用本地依赖的方式引入SDK，即手动将本地下载的SDK包引入到工程，还需要添加间接依赖：httpx、pydantic、jinja2。

```
pip install huawei_wiseagent_dev_sdk_python-{version}-py3-none-any.whl
```

初始化 SDK

在使用SDK之前，请配置鉴权参数。支持AI原生应用引擎平台的API Key鉴权。

```
import os

os.environ["WISEAGENT_API_KEY"] = "wiseagent-api-key"
```

代码样例：

```
from wiseagent_dev_sdk_python.modules.models import ModelRouter
model_router = ModelRouter()

response = model_router.chat.completion.create(
    messages=[{"role": "user", "content": "你好"}],
    model="model_id"
)
print(response["choices"][0]["message"]["content"])
```

4.4 应用示例

与文本对话模型进行交互

场景描述

提供用户与模型进行对话的能力，用户将文本内容写入到content中，选择填写合适的文本对话模型，与模型进行交互。其中环境变量WISEAGENT_API_KEY填写AI原生应用引擎的平台API KEY，该值用于鉴权，为必填。返回结果存在response中，接口返回为OpenAI的标准返回格式，可以自行解析使用。如果接口报错请参考**错误处理**。

代码示例

```
import os
os.environ["WISEAGENT_API_KEY"] = "wiseagent-api-key"

from wiseagent_dev_sdk_python.modules.models import ModelRouter
```

```
model_router = ModelRouter()

response = model_router.chat.completion.create(
    messages=[{"role": "user", "content": "你好"}],
    model="model_id"
)
print(response["choices"][0]["message"]["content"])
```

检索知识库数据

场景描述

用户检索知识库数据，根据用户提供的检索信息，返回命中的信息数据。query填写待检索内容，vector_index_id为知识库的id。其中环境变量WISEAGENT_API_KEY填写AI原生应用引擎的平台API KEY，该值用于鉴权，为必填。接口会返回检索到的切片信息。如果接口报错请参考**错误处理**。

代码示例

```
import os
os.environ["WISEAGENT_API_KEY"] = "wiseagent-api-key"

from wiseagent_dev_sdk_python.modules.knowledge_db import KnowledgeRetriever
knowledge_retriever = KnowledgeRetriever()
response = knowledge_retriever.retrieve(query="介绍一下AI原生应用引擎", vector_index_id="knowledge_id")
print(response)
```

通过提示语模板实例化提示语

场景描述

提供用户通过提示语模板实例化提示语的能力。其中template为提示语模板，支持“f-string”和“jinja2”的占位符格式。通过format方法实例化提示语，实现对提示语模板中占位符的替换，生成可供模型使用的提示语。

代码示例

```
from wiseagent_dev_sdk_python.modules.prompt import PromptTemplate
prompt = PromptTemplate.from_template(template="123{var}", template_format="f-
string").format(var="456")
print(prompt)
```

4.5 modules 模块

4.5.1 HttpxClient

实例化

表 4-3 实例化参数

参数名	参数类型	是否必选	参数描述
base_url	str	False	模型网关开放的ip和端口，有默认值可以不填。功能类似OpenAI的base_url。

参数名	参数类型	是否必选	参数描述
custom_headers	dict	False	其他自定义请求头信息。
http_client	httpx.client	False	自定义的httpx client。
kwargs	-	False	其他可选参数，如api_key、timeout等。

4.5.2 ModelRouter

ModelRouter提供了AI原生应用引擎模型网关的接口，可以访问AI原生应用引擎上的模型，接口调用类似OpenAI。

导入 ModelRouter

```
from wiseagent_dev_sdk_python.modules.models import ModelRouter
```

实例化

表 4-4 实例化参数

参数名	参数类型	是否必选	参数描述
base_url	str	False	模型网关开放的ip和端口，有默认值可以不填。功能类似OpenAI的base_url。
custom_headers	dict	False	其他自定义请求头信息。
http_client	httpx.client	False	自定义的httpx client。
kwargs	-	False	其他可选参数，如api_key、timeout等。

类属性

表 4-5 类属性

属性名称	属性类型
ModelRouter.chat	Chat object
ModelRouter.embeddings	Embeddings object
ModelRouter.images	Image object

接口调用

- **chat**

  调用大语言模型推理服务，根据用户问题，获取大语言模型的回答。

  - **接口调用**：ModelRouter.chat.completion.create
  - **参数说明**

    表 4-6 参数说明

参数名	参数类型	是否必选	参数描述
messages	List[str]	True	包含到目前为止的对话的消息列表。
model	str	True	模型服务调用唯一id字段。 平台定义了3种模型服务： • 平台预置模型服务 登录AI原生应用引擎，在左侧导航栏选择“资产中心”，选择“大模型”页签，单击模型卡片进入模型详情页面，查看模型服务调用ID。 • 租户部署模型服务 登录AI原生应用引擎，在左侧导航栏选择“模型中心 > 我的模型服务”，选择“我部署的”页签，在模型服务列表中复制模型服务调用ID。 • 租户接入模型服务 登录AI原生应用引擎，在左侧导航栏选择“模型中心 > 我的模型服务”，选择“我接入的”页签，在模型服务列表中复制模型服务调用ID。
stream	bool	False	是否流式返回。
kwargs	-	False	其他OpenAI Chat Completion参数。

  - **返回值**

    表 4-7 返回值

键	值类型	描述
id	str	文本对话唯一标识符。

键	值类型	描述
choices	list	文本生成选项列表。如果n大于1，则可以是多个。
created	int	Unix时间戳（以秒为单位）。
model	str	使用的模型。
usage	str	请求token用量。
object	str	当前为chat.completion。

- **Embedding**

  将用户输入的文本转化成数字向量，多用于从向量化知识库中查询相似的文本。

  - **接口调用**：ModelRouter.embeddings.create
  - **参数说明**

    表 4-8 参数说明

参数名	参数类型	是否必选	参数描述
input	str or List[str]	True	输入内容。
model	str	True	模型服务调用唯一id字段。 平台定义了3种模型服务： • 平台预置模型服务 登录AI原生应用引擎，在左侧导航栏选择“资产中心”，选择“大模型”页签，单击模型卡片进入模型详情页面，查看模型服务调用ID。 • 租户部署模型服务 登录AI原生应用引擎，在左侧导航栏选择“模型中心 > 我的模型服务”，选择“我部署的”页签，在模型服务列表中复制模型服务调用ID。 • 租户接入模型服务 登录AI原生应用引擎，在左侧导航栏选择“模型中心 > 我的模型服务”，选择“我接入的”页签，在模型服务列表中复制模型服务调用ID。

参数名	参数类型	是否必选	参数描述
kwargs	-	-	其他OpenAI Embedding参数。

- **返回值**

表 4-9 返回值

键	值类型	描述
data	List[Embedding]	Embedding object的列表。
model	str	模型服务调用唯一id字段。 平台定义了3种模型服务： • 平台预置模型服务 登录AI原生应用引擎，在左侧导航栏选择“资产中心”，选择“大模型”页签，单击模型卡片进入模型详情页面，查看模型服务调用ID。 • 租户部署模型服务 登录AI原生应用引擎，在左侧导航栏选择“模型中心 > 我的模型服务”，选择“我部署的”页签，在模型服务列表中复制模型服务调用ID。 • 租户接入模型服务 登录AI原生应用引擎，在左侧导航栏选择“模型中心 > 我的模型服务”，选择“我接入的”页签，在模型服务列表中复制模型服务调用ID。
usage	dict	Token消耗。
object	str	当前为list。

表 4-10 Embdding object

键	值类型	描述
index	int	Embedding索引。
embeddings	List[float]	向量，一个浮点数列表。
object	str	当前为embedding。

- **Images**
  - 将用户输入的文本转化成图片。
    - **接口调用：**ModelRouter.images.generate
    - **参数说明：**

      表 4-11 参数说明

参数名称	参数类型	是否必选	参数描述
prompt	str	True	输入的文本。
model	str	True	模型服务调用唯一id字段。平台定义了3种模型服务： • 平台预置模型服务 登录**AI原生应用引擎**，在左侧导航栏选择“资产中心”，选择“大模型”页签，单击模型卡片进入模型详情页面，查看模型服务调用ID。 • 租户部署模型服务 登录**AI原生应用引擎**，在左侧导航栏选择“模型中心 > 我的模型服务”，选择“我部署的”页签，在模型服务列表中复制模型服务调用ID。 • 租户接入模型服务 登录**AI原生应用引擎**，在左侧导航栏选择“模型中心 > 我的模型服务”，选择“我接入的”页签，在模型服务列表中复制模型服务调用ID。
kwargs	-	-	其他OpenAI Image参数，如：size。

  - 将用户输入的图片转换成文本。
    - **接口调用：**ModelRouter.images.image2text
    - **参数说明：**

表 4-12 参数说明

参数名称	参数类型	是否必选	参数描述
messages	List[str]	True	包含到目前为止的对话的消息列表。
model	str	True	模型服务调用唯一id字段。平台定义了3种模型服务： • 平台预置模型服务 登录**AI原生应用引擎**，在左侧导航栏选择“资产中心”，选择“大模型”页签，单击模型卡片进入模型详情页面，查看模型服务调用ID。 • 租户部署模型服务 登录**AI原生应用引擎**，在左侧导航栏选择“模型中心 > 我的模型服务”，选择“我部署的”页签，在模型服务列表中复制模型服务调用ID。 • 租户接入模型服务 登录**AI原生应用引擎**，在左侧导航栏选择“模型中心 > 我的模型服务”，选择“我接入的”页签，在模型服务列表中复制模型服务调用ID。
kwargs	-	-	其他大模型参数，如：temperature、top_p。

4.5.3 PromptTemplate

PromptTemplate提供了提示语编写与组装的能力，目前支持f-string和jinja两种形式的模板。

导入 PromptTemplate

```
from wiseagent_dev_sdk_python.modules.prompt import PromptTemplate
```

类属性

表 4-13 类属性

参数名	参数类型	是否必选	参数描述
input_variables	List[str]	True	提示语模板所需的变量名称列表。
template	str	True	提示语模板。
template_format	str	True	提示语模板格式，支持f-string、jinja2两种。默认f-string。
partial_variable	dict	False	格式化模板的部分变量。
validate_template	bool	False	是否尝试验证模板。

接口调用

PromptTemplate.from_template

初始化提示语模板，支持两种占位符格式以及部分变量值的预先填入。

- **参数说明**

  表 4-14 参数说明

参数名称	参数类型	是否必选	参数描述
template	str	True	提示语模板。
template_format	str	False	提示语模板格式，支持f-string、jinja2两种。默认f-string。
partial_variable	dict	False	格式化模板的部分变量。

- **返回值：**PromptTemplate object

PromptTemplate.format

实例化提示语模板。

- **参数说明**

表 4-15 参数说明

参数名称	参数类型	是否必选	参数描述
kwargs	dict	True	格式化提示语模板所需参数。

- **返回值：**提示语字符串。

4.5.4 KnowledgeRetriever

KnowledgeRetriever提供了知识库检索的能力，可以检索开发者的知识库，提取相关的信息。

导入 KnowledgeRetriever

```
from wiseagent_dev_sdk_python.modules.knowledge_db import
KnowledgeRetriever
```

实例化

表 4-16 实例化参数

参数名	参数类型	是否必选	参数描述
base_url	str	False	模型网关开放的ip和端口，有默认值可以不填。功能类似OpenAI的base_url。
custom_headers	dict	False	其他自定义请求头信息。
http_client	httpx.client	False	自定义的httpx client。
kwargs	-	False	其他可选参数，如api_key、timeout等。

接口调用

KnowledgeRetriever.retrieve

检索知识库数据，根据用户提供的检索信息，返回命中的信息数据。

参数说明

表 4-17 参数说明

参数名称	参数类型	是否必选	参数描述
vector_index_id	str	True	知识库id。 进入**AI原生应用引擎**，在左侧导航栏选择“知识中心 > 知识库”，可从页面知识库ID栏获取。
query	str	False	搜索关键字，用于检索匹配结果。 可以为null，如果不为null，字符串长度介于0到2048之间。
top_n	int	False	检索返回切片限制数量。 可以为null，如果不为null，取值大于1。
similarity_min	float	False	相似度最小值，数值越大表示相似度越高。 可以为null，如果不为null，取值介于0到1之间。
filter	**SearchSqlFilter** object	False	过滤条件。 可以为null，如果不为null，则为SearchSqlFilter类对象。
order_by	**SqlOrder** object	False	排序规则。 可以为null，如果不为null，则为SqlOrder类对象。

表 4-18 SearchSqlFilter

参数	是否必选	参数类型	描述
group_type	否	String	过滤条件运算符。 只有一个expression时，不需要group_type，group_type可以为null。 枚举值： • **AND** • **OR**
expressions	否	Array of **Expression** objects	过滤条件。 非空，条件数量介于1到10之间。

表 4-19 Expression

参数	是否必选	参数类型	描述
field	否	String	过滤字段。 非空，字符串长度介于1到100之间。
field_type	否	String	过滤字段类型。 可以为null，如果不为null，枚举值： • **INT** • **FLOAT** • **BOOLEAN** • **STRING**
operator	否	String	过滤操作符。 可以为null，如果不为null，枚举值： • **EQUAL** • **NOT_EQUAL** • **GREAT_THAN** • **GREAT_EQUAL** • **LESS_THAN** • **LESS_EQUAL** • **IN** • **NOTIN**
values	否	Array of strings	过滤值。 非空，数量介于1到100之间，每个字符串长度最大不超过2000。

表 4-20 SqlOrder

参数	是否必选	参数类型	描述
order_items	否	Array of **OrderItem** objects	排序规则。 非空，数量介于1到10之间。

表 4-21 OrderItem

参数	是否必选	参数类型	描述
field	否	String	排序字段。 非空，字符串长度介于1到100之间。
field_type	否	String	排序字段类型。 可以为null，如果不为null，枚举值： • **INT** • **FLOAT** • **BOOLEAN** • **STRING**
order_type	否	String	排序类型。 不为null，枚举值： • **ASC** • **DESC**

返回值

EmbedData对象列表。

表 4-22 EmbedData

键	值类型	描述
id	str	切片id。
document	str	切片向量化内容。
metadata	dict	文本过滤字段。
similarity	float	向量化内容（document）和检索关键字（query）的向量相似度。

4.5.5 ApplicationCenter

ApplicationCenter提供了开发者调用部署在平台的应用的能力。

导入 ApplicationCenter

```
from wiseagent_dev_sdk_python.modules.tools import ApplicationCenter
```

实例化

表 4-23 实例化参数

参数名	参数类型	是否必选	参数描述
base_url	str	False	模型网关开放的IP和端口，有默认值可以不填。功能类似OpenAI的base_url。
custom_headers	dict	False	其他自定义请求头信息。
http_client	httpx.client	False	自定义的httpx client。
kwargs	-	False	其他可选参数，如api_key、timeout等。

接口调用

ApplicationCenter.get_tools

调用用户配置的工具的执行动作。

参数说明

表 4-24 参数说明

参数名称	参数类型	是否必选	参数描述
app_id	str	True	工具的执行动作ID。 进入**AI原生应用引擎**，在左侧导航栏选择“Agent编排中心 > 我的工具”，在工具列表中复制执行动作ID。

参数名称	参数类型	是否必选	参数描述
body	dict	True	请求体。 结构与工具的执行动作的配置相关，并且所有请求头中的入参与请求参数均添加至请求体中，由AI原生应用引擎自动完成分配。如果为GET请求则为非必填，如果为POST请求则为必填。 比如工作流配置了query_param作为查询参数，header_param作为请求头参数，body_param_1与body_param_2作为请求体参数，此时调用本接口只需要将这些参数依次传入，AI原生应用引擎自动按照名称进行分配，并完成工具的执行动作的调用。 具体结构请参照本接口的请求实例。
headers	dict	False	请求头。
stream	bool	False	是否流式返回（需要应用本身支持）。

返回值

应用返回。

4.6 错误处理

在使用AI原生应用引擎SDK时，当服务端或者SDK端出错时，SDK会返回相应的异常信息，请参见下表进行处理。

状态码	错误码	错误信息	描述	处理措施
200	AIAE.22001001	API调用异常	API调用异常	调用接口url、请求方式错误或出现访问其他用户资源的越权问题，请检查后重试
200	AIAE.22001002	IAM认证异常	IAM认证异常	后端服务错误，请联系技术支持
200	AIAE.22001003	认证失败:{reason}	认证失败:{reason}	请参考返回的error message，或联系技术支持

状态码	错误码	错误信息	描述	处理措施
200	AIAE.22001004	参数{parameterName}异常	参数{parameterName}异常	输入的参数有误，请参考返回的error message进行修改后重试
200	AIAE.22001005	{type}类型远程调用失败，错误信息为{error_msg}	{type}类型远程调用失败，错误信息为{error_msg}	请参考返回的error message处理后重试
200	AIAE.22001006	文件上传失败:{reason}	文件上传失败:{reason}	文件上传失败，请参考返回的error message处理后重试
200	AIAE.22001007	技能未设置鉴权信息	技能未设置鉴权信息	在wiseAgent页面为技能设置鉴权信息。
200	AIAE.22001008	用户无权限访问当前资源	用户无权限访问当前资源	请更换用户后访问
200	AIAE.22001009	开启的流无法被删除	开启的流无法被删除	关闭流后重试
200	AIAE.22009001	系统内部错误，请联系管理员	系统内部错误，请联系管理员	系统内部错误，请联系技术支持
400	AIAE.31001106	AK/SK signature verify failed, please check and try again later!	很抱歉，AK/SK签名验证失败!	很抱歉，AK/SK签名验证失败!
400	AIAE.31001601	Sensitive request error, please try again later!	很抱歉，请求内容中包含敏感信息，请重试!	很抱歉，请求内容中包含敏感信息，请重试!
400	AIAE.31001602	Sensitive response error, please try again later!	很抱歉，返回内容中包含敏感信息，请重试!	很抱歉，返回内容中包含敏感信息，请重试!
400	AIAE.31001603	Sensitive content error, please try again later!	很抱歉，请求或返回内容中包含敏感信息，请重试!	很抱歉，请求或返回内容中包含敏感信息，请重试!

状态码	错误码	错误信息	描述	处理措施
400	AIAE.31001701	Bad request parameter error, please check and try again later!	很抱歉，请求参数异常，请检查后重试！	很抱歉，请求参数异常，请检查后重试！
400	AIAE.40001003	Authentication failed	X-Auth-Token鉴权失败	很抱歉，X-Auth-Token 鉴权失败
400	AIAE.40002605	knowledgeBase status is not ENABLE	很抱歉，知识库未启用，没有权限查询	很抱歉，知识库未启用，没有权限查询
401	AIAE.31001101	User not login, please check and try again later!	很抱歉，用户未登录，请登录后重试！	很抱歉，用户未登录，请登录后重试！
401	AIAE.31001102	AK/SK verify failed, please check and try again later!	很抱歉，AK/SK校验失败！	很抱歉，AK/SK校验失败！
401	AIAE.31001103	Authentication verify failed, please check and try again later!	很抱歉，鉴权失败！	很抱歉，鉴权失败！
401	AIAE.31001104	API Key verify failed, please check and try again later!	很抱歉，API Key校验失败！	很抱歉，API Key校验失败！
401	AIAE.31001201	Tenant id is empty, please check and try again later!	很抱歉，空的租户id，请检查后重试！	很抱歉，空的租户id，请检查后重试！
401	AIAE.31005001	The third model service authentication is abnormal, please check and try again later!	很抱歉，三方模型服务鉴权异常，请检查您的鉴权信息！	很抱歉，三方模型服务鉴权异常，请检查您的鉴权信息！
401	AIAE.31005002	Invalid third api key, please check and try again later!	很抱歉，三方模型服务鉴权API Key异常，请检查您的鉴权信息！	很抱歉，三方模型服务鉴权API Key异常，请检查您的鉴权信息！

状态码	错误码	错误信息	描述	处理措施
401	AIAE.31005007	The third model service authentication is empty, please set and try again later!	很抱歉，三方模型服务鉴权未设置，请设置后重试！	很抱歉，三方模型服务鉴权未设置，请设置后重试！
402	AIAE.31005005	The third model service exceeded current quota error, please check and try again later!	很抱歉，账户异常，请检查您的账户余额！	很抱歉，账户异常，请检查您的账户余额！
403	AIAE.31001105	Role permission verify failed, please check and try again later!	很抱歉，当前用户不允许该操作！	很抱歉，当前用户不允许该操作！
403	AIAE.31001501	SKU not subscribed to model service, please try again after subscribed!	很抱歉，您未订阅当前模型服务的SKU，请联系管理员订阅！	很抱歉，您未订阅当前模型服务的SKU，请联系管理员订阅！
403	AIAE.31001502	SKU verify failed, please check and try again later!	很抱歉，SKU校验异常，请检查您的SKU！	很抱歉，SKU校验异常，请检查您的SKU！
403	AIAE.40001004	User does not have permission	当前用户没有权限	很抱歉，当前用户没有权限
404	AIAE.31001202	Model not published, please try again after model published!	很抱歉，模型服务未发布，请发布后重试！	很抱歉，模型服务未发布，请发布后重试！
404	AIAE.31001702	Model not exists, please check and try again later!	很抱歉，模型服务不存在，请检查您输入的模型服务名称！	很抱歉，模型服务不存在，请检查您输入的模型服务名称！

状态码	错误码	错误信息	描述	处理措施
408	AIAE.31001003	Connection timeout, please try again later!	很抱歉，网络连接超时，请稍后重试！	很抱歉，网络连接超时，请稍后重试！
408	AIAE.31005006	The third model service connect timeout, please try again later!	很抱歉，三方服务连接超时，请稍后重试！	很抱歉，三方服务连接超时，请稍后重试！
429	AIAE.31001002	Request too frequent error, please try again later!	很抱歉，您的请求过于频繁，请稍后重试！	很抱歉，您的请求过于频繁，请稍后重试！
429	AIAE.31005003	The third model service rate limit exceeded, please try again later!	很抱歉，您的请求当前已达最大并发数，请稍后重试！	很抱歉，您的请求当前已达最大并发数，请稍后重试！
429	AIAE.31005004	The third model service overload error, please try again later!	很抱歉，服务当前超载，请稍后重试！	很抱歉，服务当前超载，请稍后重试！
500	AIAE.31001001	Internal server error, please try again later!	很抱歉，服务内部出现了问题，请稍后重试！	很抱歉，服务内部出现了问题，请稍后重试！
500	AIAE.31001004	Ai security governance service error, please try again later or disable ai security governance service!	很抱歉，内容审核服务出现了问题，请稍后重试！	很抱歉，内容审核服务出现了问题，请稍后重试！
500	AIAE.31005000	Invalid third response, please try again later!	很抱歉，调用三方模型服务异常，请稍后重试！	很抱歉，调用三方模型服务异常，请稍后重试！
400	UniModel.Request.0001	请求参数错误	模型服务的请求参数错误	检查模型的请求参数

状态码	错误码	错误信息	描述	处理措施
500	UniDataEmbed.Internal.0001	请求失败	请求向量化服务失败	检查向量知识库配置
500	UniModel.Internal.0001	模型访问失败	无法访问选择的模型	检查模型是否已经正常部署
500	UniModel.Internal.0002	模型返回超时	模型服务返回超时	检查网络情况，或者减少模型返回内容
500	WS.00100001	AUTHENTICATION_ERROR	鉴权错误	检查访问权限
500	WS.00100002	SHA_ERROR	SHA算法错误	检查签名所用的算法
500	WS.00100003	SIGN_ERROR	请求签名错误	检查请求签名
500	WS.00100005	NO_ACCESS_ERROR	无访问权限错误	检查接口访问权限

4.7 日志处理

日志级别配置

日志级别通过环境变量WISEAGENT_LOG配置：

```
import os
os.environ["WISEAGENT_LOG"] = "INFO"  # 配置日志级别为INFO
```

日志级别默认为WARNING，支持配置为DEBUG、INFO。

日志格式

日志格式：

```
[%(asctime)s - %(name)s:%(lineno)d - %(levelname)s] %(message)s
```

- asctime：日志打印时间
- name：打印日志的文件名
- lineno：哪一行代码打印的日志
- levelname：日志级别
- message：具体日志内容

示例如下：

[2024-12-10 19:21:28 - wiseagent_dev_sdk_python.modules.models.chat:31 - INFO] Request base url is https://xxxxx, path is /v1/chat/completions