Apusic 金蝶天燕

生成时间：2026-07-13

API开放平台手册

金蝶Apusic分布式配置中心 for Apollo V1.0

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。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。

目录

1 前言

本文档为金蝶Apusic分布式配置中心for Apollo（ADCC for Apollo）V1.0的API开放平台手册，是基于配置中心内核与web管控能力对外开放的标准化接口集成文档。手册定义了配置中心全量开放API的调用规范、鉴权方式、请求参数、返回格式及异常处理规则，覆盖配置管理、环境集群管控、权限操作、变更审计等核心能力。

1.1 适用对象

本文档适用于第三方系统集成开发人员，支持外部平台、运维系统通过标准API对接分布式配置中心，实现自动化管控、跨系统联动集成。

1.2 相关文档

了解更多ADCC for Apollo V1.0产品相关的信息，请参阅以下ADCC for Apollo V1.0产品手册文档集：

序号	手册文档	说明
1	金蝶Apusic分布式配置中心for Apollo V1.0 安装手册	详细介绍如何在各操作系统上安装ADCC for Apollo，以及服务组件实例启停操作，产品的注册过程。
2	金蝶Apusic分布式配置中心for Apollo V1.0 用户手册	详细介绍 ADCC 相关功能的使用、配置、管理及配套工具的使用方法。
3	金蝶Apusic分布式配置中心for Apollo V1.0 开发手册	详细介绍基于各种开发语言进行ADCC for Apollo客户端应用开发的说明。
4	金蝶Apusic分布式配置中心for Apollo V1.0 API开放平台手册	详细介绍ADCC for Apollo的标准API。

1.3 技术支持

ADCC for Apollo产品提供全面的技术支持服务，您可以通过以下方式获得技术支持：

- 网址：www.apusic.com
- 电话：400-855-5800
- 邮箱：support@apusic.com
- 金蝶云社区：https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN

您在取得技术支持时，请提供如下信息：

- 您的姓名
- 公司信息与联系方式
- 操作系统及其版本
- 产品版本号

- 出现异常及错误的日志、截图等详细信息

2 什么是API开放平台

ADCC for Apollo提供了一套的Http REST接口，使第三方应用能够自己管理配置。虽然分布式配置中心本身提供了Portal来管理配置，但是在有些情景下，应用需要通过程序去管理配置。

3 第三方应用接入API开放平台

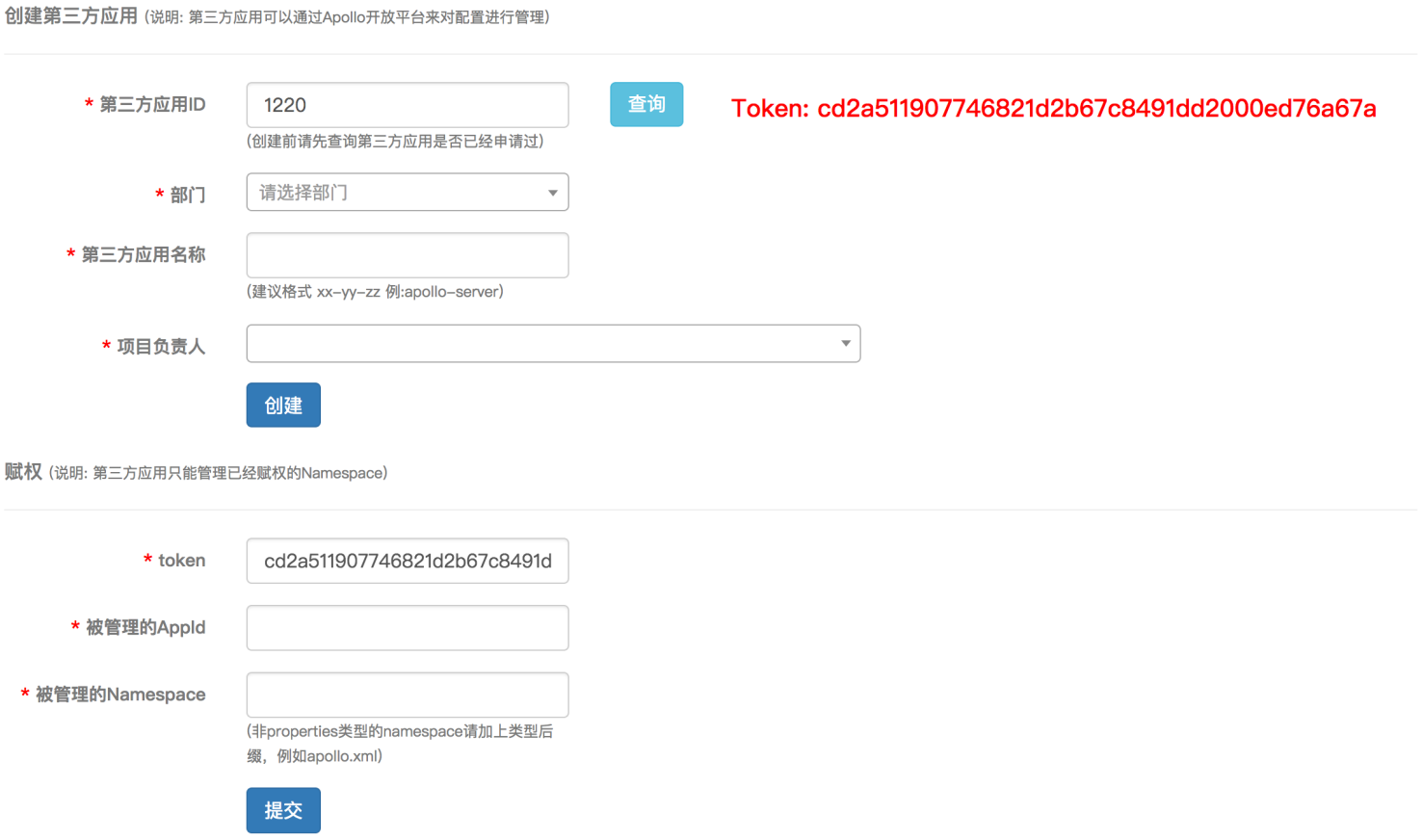
3.1 注册第三方应用

第三方应用负责人需要向管理员提供一些第三方应用基本信息。

基本信息如下：

- 第三方应用的AppId、应用名、部门
- 第三方应用负责人

Apollo管理员在 `http://{portal_address}/open/add-consumer.html` 创建第三方应用，创建之前最好先查询此AppId是否已经创建。创建成功之后会生成一个token，如下图所示：



3.2 查看第三方应用

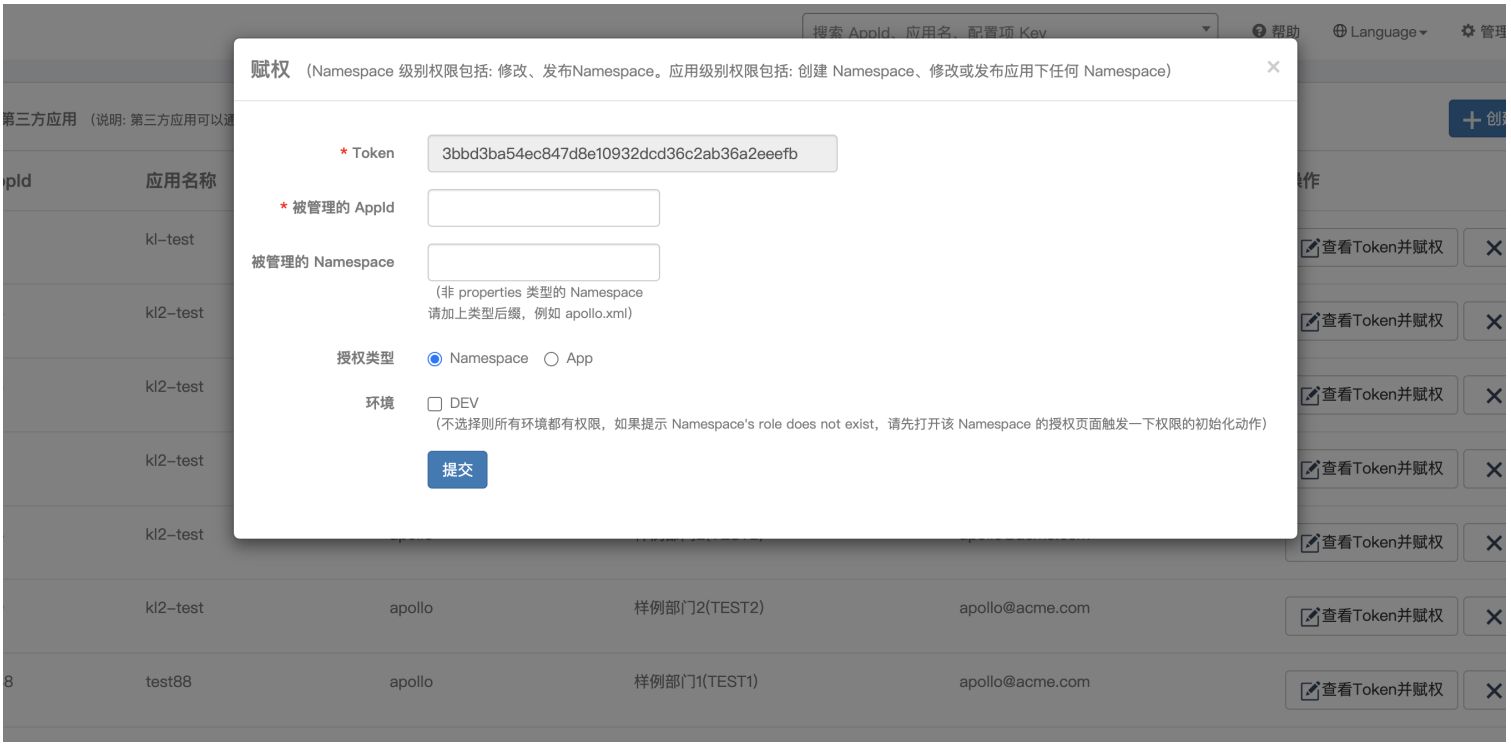
Apollo管理员在 `http://{portal_address}/open/manage.html` 页面可以查看第三方应用列表。并提供了【查看Token并赋权】、【删除】等管理操作，如下图所示：

创建第三方应用 （说明: 第三方应用可以通过 Apollo 开放平台来对配置进行管理）

+ 创建应用

AppId	应用名称	负责人	部门	邮箱	操作
kl	kl-test	apollo	样例部门1(TEST1)	apollo@acme.com	查看Token并赋权 删除
kl5	kl2-test	apollo	样例部门2(TEST2)	apollo@acme.com	查看Token并赋权 删除
kl6	kl2-test	apollo	样例部门2(TEST2)	apollo@acme.com	查看Token并赋权 删除
kl7	kl2-test	apollo	样例部门2(TEST2)	apollo@acme.com	查看Token并赋权 删除
kl8	kl2-test	apollo	样例部门2(TEST2)	apollo@acme.com	查看Token并赋权 删除
kl9	kl2-test	apollo	样例部门2(TEST2)	apollo@acme.com	查看Token并赋权 删除
kl88	test88	apollo	样例部门1(TEST1)	apollo@acme.com	查看Token并赋权 删除
kkkkk	kkk-test	apollo	样例部门1(TEST1)	apollo@acme.com	查看Token并赋权 删除
store-rcp	store	apollo	样例部门1(TEST1)	apollo@acme.com	查看Token并赋权 删除

【查看Token并赋权】的模态框页面如下图所示：



3.3 给已注册的第三方应用授权

第三方应用不应该能操作任何Namespace的配置，所以需要给token绑定可以操作的Namespace。Apollo管理员在 `http://{portal_address}/open/add-consumer.html` 页面给token赋权。赋权之后，第三方应用就可以通过Apollo提供的Http REST接口来管理已授权的Namespace的配置了。

3.4 第三方应用调用Apollo Open API

3.4.1 调用Http REST接口

任何语言的第三方应用都可以调用Apollo的Open API，在调用接口时，需要设置注意以下两点：

- Http Header中增加一个Authorization字段，字段值为申请的token
- Http Header的Content-Type字段需要设置成application/json;charset=UTF-8

3.4.2 Java应用通过apollo-openapi调用Apollo Open API

从1.1.0版本开始，Apollo提供了apollo-openapi客户端，所以Java语言的第三方应用可以更方便地调用Apollo Open API。

首先引入 `apollo-openapi` 依赖：

```
<dependency>
    <groupId>com.ctrip.framework.apollo</groupId>
    <artifactId>apollo-openapi</artifactId>
    <version>1.7.0</version>
</dependency>
```

在程序中构造 `ApolloOpenApiClient` ：

```
String portalUrl = "http://localhost:8070"; // portal url
String token = "e16e5cd903fd0c97a116c873b448544b9d086de9"; // 申请的token
ApolloOpenApiClient client = ApolloOpenApiClient.newBuilder()
                                                .withPortalUrl(portalUrl)
                                                .withToken(token)
                                                .build();
```

后续就可以通过 `ApolloOpenApiClient` 的接口直接操作Apollo Open API了，接口说明参见下面的Rest接口文档。

3.4.3 .Net core应用调用Apollo Open API

.Net core也提供了open api的客户端，详见https://github.com/ctripcorp/apollo.net/pull/77

3.4.4 Shell Scripts调用Apollo Open API

封装了bash的function，底层使用curl来发送HTTP请求

- bash函数：openapi.sh
- 使用示例：openapi-usage-example.sh
- 全部和openapi有关的shell脚本在文件夹 https://github.com/apolloconfig/apollo/tree/master/scripts/sql 下

4 接口文档

4.1 URL路径参数说明

参数名	参数说明
env	所管理的配置环境
appId	所管理的配置AppId
clusterName	所管理的配置集群名， 一般情况下传入 default 即可。如果是特殊集群，传入相应集群的名称即可
namespaceName	所管理的Namespace的名称，如果是非properties格式，需要加上后缀名，如`sample.yml`

4.2 API接口列表

4.2.1 获取App的环境，集群信息

- URL : http://{portal_address}/openapi/v1/apps/{appId}/envclusters
- Method : GET
- Request Params : 无
- 返回值Sample:

```
[
    {
        "env":"FAT",
        "clusters":[ //集群列表
            "default",
            "FAT381"
        ]
    },
    {
        "env":"UAT",
        "clusters":[
            "default"
        ]
    },
    {
        "env":"PRO",
        "clusters":[
            "default",
            "SHAOY",
            "SHAJQ"
        ]
```

```
    }
]
```

4.2.2 获取App信息

- URL : http://{portal_address}/openapi/v1/apps
- Method : GET
- Request Params :

参数名	必选	类型	说明
appIds	false	String	appId列表，以逗号分隔，如果为空则返回所有App信息

- 返回值Sample:

```
[
    {
        "name":"first_app",
        "appId":"100003171",
        "orgId":"development",
        "orgName":"研发部",
        "ownerName":"apollo",
        "ownerEmail":"test@test.com",
        "dataChangeCreatedBy":"apollo",
        "dataChangeLastModifiedBy":"apollo",
        "dataChangeCreatedTime":"2019-05-08T09:13:31.000+0800",
        "dataChangeLastModifiedTime":"2019-05-08T09:13:31.000+0800"
    },
    {
        "name":"apollo-demo",
        "appId":"100004458",
        "orgId":"development",
        "orgName":"产品研发部",
        "ownerName":"apollo",
        "ownerEmail":"apollo@cmcm.com",
        "dataChangeCreatedBy":"apollo",
        "dataChangeLastModifiedBy":"apollo",
        "dataChangeCreatedTime":"2018-12-23T12:35:16.000+0800",
        "dataChangeLastModifiedTime":"2019-04-08T13:58:36.000+0800"
    }
]
```

4.2.3 获取集群接口

- URL： http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters/{clusterName}
- Method： GET
- Request Params： 无
- 返回值Sample:

```
{
    "name":"default",
    "appId":"100004458",
    "dataChangeCreatedBy":"apollo",
    "dataChangeLastModifiedBy":"apollo",
    "dataChangeCreatedTime":"2018-12-23T12:35:16.000+0800",
    "dataChangeLastModifiedTime":"2018-12-23T12:35:16.000+0800"
}
```

4.2.4 创建集群接口

可以通过此接口创建集群，调用此接口需要授予第三方APP对目标APP的管理权限。

- URL： http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters
- Method： POST
- Request Params： 无
- 请求内容(Request Body, JSON格式)：

参数名	必选	类型	说明
name	true	String	Cluster的名字
appId	true	String	Cluster所属的AppId
dataChangeCreatedBy	true	String	namespace的创建人，格式为域账号，也就是sso系统的User ID

- 返回值 Sample：

```
 {
    "name":"someClusterName",
    "appId":"100004458",
    "dataChangeCreatedBy":"apollo",
    "dataChangeLastModifiedBy":"apollo",
    "dataChangeCreatedTime":"2018-12-23T12:35:16.000+0800",
    "dataChangeLastModifiedTime":"2018-12-23T12:35:16.000+0800"
}
```

4.2.5 获取集群下所有Namespace信息接口

- URL : http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters/{clusterName}/namespaces
- Method: GET

- Request Params: 无
- 返回值Sample:

```
[
  {
    "appId": "100003171",
    "clusterName": "default",
    "namespaceName": "application",
    "comment": "default app namespace",
    "format": "properties", //Namespace格式可能取值为: properties、xml、json、yml、
yaml
    "isPublic": false, //是否为公共的Namespace
    "items": [ // Namespace下所有的配置集合
      {
        "key": "batch",
        "value": "100",
        "dataChangeCreatedBy": "song_s",
        "dataChangeLastModifiedBy": "song_s",
        "dataChangeCreatedTime": "2016-07-21T16:03:43.000+0800",
        "dataChangeLastModifiedTime": "2016-07-21T16:03:43.000+0800"
      }
    ],
    "dataChangeCreatedBy": "song_s",
    "dataChangeLastModifiedBy": "song_s",
    "dataChangeCreatedTime": "2016-07-20T14:05:58.000+0800",
    "dataChangeLastModifiedTime": "2016-07-20T14:05:58.000+0800"
  },
  {
    "appId": "100003171",
    "clusterName": "default",
    "namespaceName": "FX.apollo",
    "comment": "apollo public namespace",
    "format": "properties",
    "isPublic": true,
    "items": [
      {
        "key": "request.timeout",
        "value": "3000",
        "comment": "",
        "dataChangeCreatedBy": "song_s",
        "dataChangeLastModifiedBy": "song_s",
        "dataChangeCreatedTime": "2016-07-20T14:08:30.000+0800",
```

```
        "dataChangeLastModifiedTime": "2016-08-01T13:56:25.000+0800"
      },
      {
        "id": 1116,
        "key": "batch",
        "value": "3000",
        "comment": "",
        "dataChangeCreatedBy": "song_s",
        "dataChangeLastModifiedBy": "song_s",
        "dataChangeCreatedTime": "2016-07-28T15:13:42.000+0800",
        "dataChangeLastModifiedTime": "2016-08-01T13:51:00.000+0800"
      }
    ],
    "dataChangeCreatedBy": "song_s",
    "dataChangeLastModifiedBy": "song_s",
    "dataChangeCreatedTime": "2016-07-20T14:08:13.000+0800",
    "dataChangeLastModifiedTime": "2016-07-20T14:08:13.000+0800"
  }
]
```


4.2.6 获取某个Namespace信息接口

- URL： http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters/{clusterName}/namespaces/{namespaceName}
- Method： GET
- Request Params：无
- 返回值Sample：


```
{
    "appId": "100003171",
    "clusterName": "default",
    "namespaceName": "application",
    "comment": "default app namespace",
    "format": "properties", //Namespace格式可能取值为：properties、xml、json、yml、
yaml
    "isPublic": false, //是否为公共的Namespace
    "items": [ // Namespace下所有的配置集合
      {
        "key": "batch",
        "value": "100",
        "dataChangeCreatedBy": "song_s",
        "dataChangeLastModifiedBy": "song_s",
        "dataChangeCreatedTime": "2016-07-21T16:03:43.000+0800",
```

```
        "dataChangeLastModifiedTime": "2016-07-21T16:03:43.000+0800"
      }
    ],
    "dataChangeCreatedBy": "song_s",
    "dataChangeLastModifiedBy": "song_s",
    "dataChangeCreatedTime": "2016-07-20T14:05:58.000+0800",
    "dataChangeLastModifiedTime": "2016-07-20T14:05:58.000+0800"
  }
```

4.2.7 创建Namespace

可以通过此接口创建Namespace，调用此接口需要授予第三方APP对目标APP的管理权限。

- URL： http://{portal_address}/openapi/v1/apps/{appId}/appnamespaces
- Method： POST
- Request Params：无
- 请求内容(Request Body, JSON格式)：

参数名	必选	类型	说明
name	true	String	Namespace的名字
appId	true	String	Namespace所属的AppId
format	true	String	Namespace的格式，只能是以下类型： properties、xml、json、yml、yaml
isPublic	true	boolean	是否是公共文件
comment	false	String	Namespace说明
dataChangeCreatedBy	true	String	namespace的创建人，格式为域账号，也就是sso系统的User ID

- 返回值 Sample：

```
{
  "name": "FX.public-0420-11",
  "appId": "100003173",
  "format": "properties",
  "isPublic": true,
  "comment": "test",
  "dataChangeCreatedBy": "zhanglea",
  "dataChangeLastModifiedBy": "zhanglea",
  "dataChangeCreatedTime": "2017-04-20T18:25:49.033+0800",
  "dataChangeLastModifiedTime": "2017-04-20T18:25:49.033+0800"
}
```

- 返回值说明：

> *如果是properties文件，name = ${appId所属的部门}.${传入的name值}，例如调用接口传入的name=xy-z, format=properties，应用的部门为框架（FX）,那么name=FX.xy-z*

> *如果不是properties文件 name = ${appId所属的部门}.${传入的name值}.${format}，例如调用接口传入的name=xy-z, format=json，应用的部门为框架（FX）,那么name=FX.xy-z.json*

4.2.8 获取某个Namespace当前编辑人接口

Apollo在生产环境（PRO）有限制规则：每次发布只能有一个人编辑配置，且该次发布的人不能是该次发布的编辑人。 也就是说如果一个用户A修改了某个namespace的配置，那么在这个namespace发布前，只能由A修改，其它用户无法修改。同时，该用户A无法发布自己修改的配置，必须找另一个有发布权限的人操作。 这个接口就是用来获取当前namespace是否有人锁定的接口。在非生产环境（FAT、UAT），该接口始终返回没有人锁定。

- URL：
  http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters/{clusterName}/namespaces/{namespaceName}/lock
- Method： GET
- Request Params：无
- 返回值 Sample（未锁定）：

```
{
  "namespaceName": "application",
  "isLocked": false
}
```

- 返回值Sample(被锁定)：

```
{
  "namespaceName": "application",
  "isLocked": true,
  "lockedBy": "song_s" //锁owner
}
```

4.2.9 读取配置接口

- URL：
  http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters/{clusterName}/namespaces/{namespaceName}/items/{key}
- Method： GET
- Request Params：无
- 返回值Sample：

```
{
  "key": "timeout",
  "value": "3000",
  "comment": "超时时间",
```

```
    "dataChangeCreatedBy": "zhanglea",
    "dataChangeLastModifiedBy": "zhanglea",
    "dataChangeCreatedTime": "2016-08-11T12:06:41.818+0800",
    "dataChangeLastModifiedTime": "2016-08-11T12:06:41.818+0800"
}
```

4.2.10 新增配置接口

- URL：
  http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters/{clusterName}/namespaces/{namespaceName}/items
- Method： POST
- Request Params：无
- 请求内容(Request Body, JSON格式)：

参数名	必选	类型	说明
key	true	String	配置的key，长度不能超过128个字符。非properties格式，key固定为`content`
value	true	String	配置的value，长度不能超过20000个字符，非properties格式，value为文件全部内容
comment	false	String	配置的备注,长度不能超过256个字符
dataChangeCreatedBy	true	String	item的创建人，格式为域账号，也就是sso系统的User ID

- Request body sample：

```
{
    "key":"timeout",
    "value":"3000",
    "comment":"超时时间",
    "dataChangeCreatedBy":"zhanglea"
}
```

- 返回值Sample：

```
  {
    "key": "timeout",
    "value": "3000",
    "comment": "超时时间",
    "dataChangeCreatedBy": "zhanglea",
    "dataChangeLastModifiedBy": "zhanglea",
    "dataChangeCreatedTime": "2016-08-11T12:06:41.818+0800",
    "dataChangeLastModifiedTime": "2016-08-11T12:06:41.818+0800"
}
```

4.2.11 修改配置接口

- URL：
  http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters/{clusterName}/namespaces/{namespaceName}/items/{key}
- Method： PUT
- Request Params：

参数名	必选	类型	说明
createIfNotExists	false	Boolean	当配置不存在时是否自动创建

- 请求内容(Request Body, JSON格式)：

参数名	必选	类型	说明
key	true	String	配置的key，需和url中的key值一致。非properties格式，key固定为content
value	true	String	配置的value，长度不能超过20000个字符，非properties格式，value为文件全部内容
comment	false	String	配置的备注,长度不能超过256个字符
dataChangeLastModifiedBy	true	String	item的修改人，格式为域账号，也就是sso系统的User ID
dataChangeCreatedBy	false	String	当createIfNotExists为true时必选。item的创建人，格式为域账号，也就是sso系统的User ID

- Request body sample：

```
{
    "key":"timeout",
    "value":"3000",
    "comment":"超时时间",
    "dataChangeLastModifiedBy":"zhanglea"
}
```

- 返回值： 无

4.2.12 删除配置接口

- URL：
  http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters/{clusterName}/namespaces/{namespaceName}/items/{key}?operator={operator}
- Method： DELETE
- Request Params：

参数名	必选	类型	说明
key	true	String	配置的key。非properties格式，key固定为content
operator	true	String	删除配置的操作者，域账号

- 返回值： 无

4.2.13 发布配置接口

- URL：
  http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters/{clusterName}/namespaces/{namespaceName}/releases
- Method： POST
- Request Params：无
- Request Body：

参数名	必选	类型	说明
releaseTitle	true	String	此次发布的标题，长度不能超过64个字符
releaseComment	false	String	发布的备注，长度不能超过256个字符
releasedBy	true	String	发布人，域账号，注意：如果ApolloConfigDB.ServerConfig中的namespace.lock.switch设置为true的话（默认是false），那么该环境不允许发布人和编辑人为同一人。所以如果编辑人是zhanglea，发布人就不能再是zhanglea。

- Request Body example：

```
{
    "releaseTitle":"2016-08-11",
    "releaseComment":"修改timeout值",
    "releasedBy":"zhanglea"
}
```

- 返回值Sample：

```
{
    "appId": "test-0620-01",
    "clusterName": "test",
    "namespaceName": "application",
    "name": "2016-08-11",
    "configurations": {
        "timeout": "3000",
    },
    "comment": "修改timeout值",
    "dataChangeCreatedBy": "zhanglea",
    "dataChangeLastModifiedBy": "zhanglea",
    "dataChangeCreatedTime": "2016-08-11T14:03:46.232+0800",
    "dataChangeLastModifiedTime": "2016-08-11T14:03:46.235+0800"
}
```

4.2.14 获取某个Namespace当前生效的已发布配置接口

- URL：
  http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters/{clusterName}/namespaces/{namespaceName}/releases/latest
- Method： GET
- Request Params：无
- 返回值Sample：

```
{
    "appId": "test-0620-01",
    "clusterName": "test",
    "namespaceName": "application",
    "name": "2016-08-11",
    "configurations": {
        "timeout": "3000",
    },
    "comment": "修改timeout值",
    "dataChangeCreatedBy": "zhanglea",
    "dataChangeLastModifiedBy": "zhanglea",
    "dataChangeCreatedTime": "2016-08-11T14:03:46.232+0800",
    "dataChangeLastModifiedTime": "2016-08-11T14:03:46.235+0800"
}
```

4.2.15 回滚已发布配置接口

- URL： http://{portal_address}/openapi/v1/envs/{env}/releases/{releaseId}/rollback
- Method： PUT
- Request Params：

参数名	必选	类型	说明
operator	true	String	删除配置的操作者，域账号

- 返回值： 无

4.2.16 分页获取配置项接口

- URL：
  http://{portal_address}/openapi/v1/envs/{env}/apps/{appId}/clusters/{clusterName}/namespaces/{namespaceName}/items
- Method： GET
- Version： >= 2.1.0
- Request Params：

参数名	必选	类型	说明
page	false	int	页码，从 0 开始，默认为 0
size	false	int	页大小，默认为 50

- 返回值Sample：


```
{
    "content": [
        {
            "key": "timeout",
            "value": "3000",
            "comment": "超时时间",
            "dataChangeCreatedBy": "mghio",
            "dataChangeLastModifiedBy": "mghio",
            "dataChangeCreatedTime": "2022-07-17T21:37:41.818+0800",
            "dataChangeLastModifiedTime": "2022-07-17T21:37:41.818+0800"
        },
        {
            "key": "page.size",
            "value": "200",
            "comment": "页大小",
            "dataChangeCreatedBy": "mghio",
            "dataChangeLastModifiedBy": "mghio",
            "dataChangeCreatedTime": "2022-07-17T21:37:41.818+0800",
            "dataChangeLastModifiedTime": "2022-07-17T21:37:41.818+0800"
        }
    ],
    "page": 0,
    "size": 50,
    "total": 2
}
```

5 错误码说明

正常情况下，接口返回的Http状态码是200，下面列举了Apollo会返回的非200错误码说明。

5.1 400 - Bad Request

客户端传入参数的错误，如操作人不存在，namespace不存在等等，客户端需要根据提示信息检查对应的参数是否正确。

5.2 401 - Unauthorized

接口传入的token非法或者已过期，客户端需要检查token是否传入正确。

5.3 403 - Forbidden

接口要访问的资源未得到授权，比如只授权了对A应用下Namespace的管理权限，但是却尝试管理B应用下的配置。

5.4 404 - Not Found

接口要访问的资源不存在，一般是URL或URL的参数错误。

5.5 405 - Method Not Allowed

接口访问的Method不正确，比如应该使用POST的接口使用了GET访问等，客户端需要检查接口访问方式是否正确。

5.6 500 - Internal Server Error

其它类型的错误默认都会返回500，对这类错误如果应用无法根据提示信息找到原因的话，可以找Apollo研发团队一起排查问题。

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com