

数据服务开放网关使用说明书

1. 产品介绍	2
2. 产品架构	2
2.1. 应用架构视图	2
2.2. 部署架构视图	3
2.3. 硬件需求	3
3. 产品安装	4
3.1. 环境要求	4
3.2. 软件拷贝	4
3.3. 安装配置	5
3.4. 执行安装	6
4. 产品使用	6
4.1. 首页	6
4.2. 安全管理	7
4.2.1. 角色管理	7
4.2.2. 项目管理	7
4.2.3. 用户管理	7
4.2.4. 授权管理	8
4.3. 版本管理	8
4.3.1. 版本发布	9
4.3.2. 版本切换	9
4.4. 管理 API	10
4.4.1. 集群管理	10
4.4.2. API 管理	10
4.4.3. 流量控制	12
4.5. 设计 API	13
4.6. 系统参数设置	13
5. 接口说明	13
5.1. 获取 token 接口	13
5.2. 使用 token 方法	14

1. 产品介绍

企业数据应用服务中，会有来自各个业务系统的数据供给需求，它们可能来自 web 前端、Android、IOS、微信小程序，也可能来自外部系统的数据对接。数据接口需求通常是 REST API 的形式。但内部各个系统通常各成体系，有自己的技术框架和开发语言，存量系统研发上线在先，自研和外采的系统兼而有之，接口不可能做到提前统筹规划一致，甚至还有冲突的情况出现。

如何在做到不变更现有接口的前提下，既能提供一套统一的数据服务入口，又能避免内部接口直接暴露带来的安全风险、流量爆发等问题，把各种风险在到达内部系统之前做到有效隔离，同时高效可控。

数据服务网关产品作为一个协调者，屏蔽网关之下的数据接口实体，提供数据服务注册发现、授权和访问控制、流量控制、负载均衡、服务版本发布等能力，主要功能包括：

1. 隔离调用方和服务方
2. 统一的安全管理、访问控制、灵活的期限设置
3. 数据服务发现、注册、路由
4. 实现数据服务负载均衡、动态调整负载策略
5. 基于项目的接口发布、批量上线，版本切换
6. 在线接口编辑、发布、验证、沙箱测试

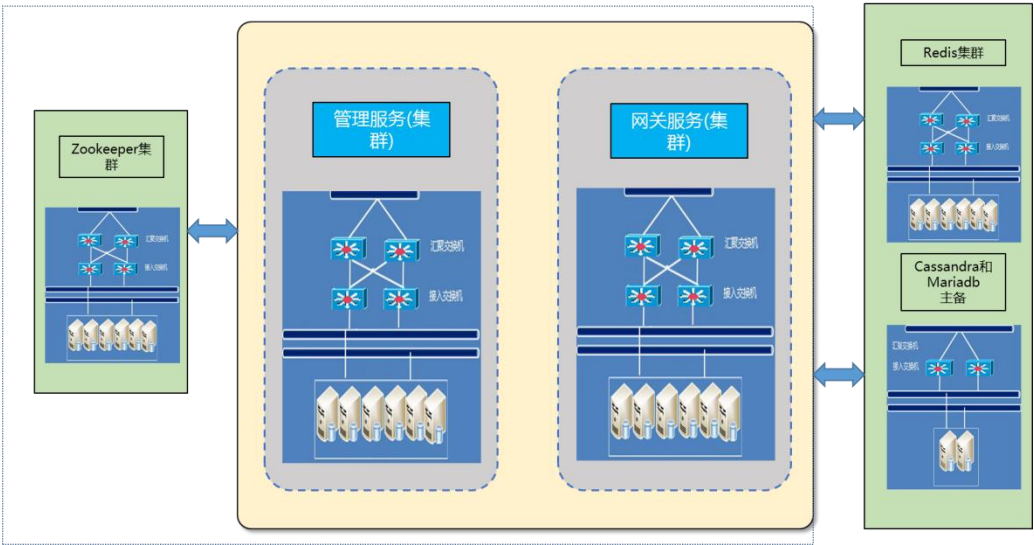
2. 产品架构

2.1. 应用架构视图



2.2. 部署架构视图

物理部署图



2.3. 硬件需求

应用	内存	硬盘容量	CPU	备注
网关管理	4G	80G	1 核	集群，功能包括用户管理、API 版本控制、

				API 状态查询、 API 流量控制、 API 网关配置、 文档与测试
redis	4G	160G	2 核	集群
网关服务	4G	80G	2 核	集群，功能包括 API 发现/注册、API 路由管理、API 服务发布、API 服务限流和降级、API 授权和访问管理、API 调用日志采集查询监控
cassandra	4G	160G	2 核	主备
mysql	4G	160G	2 核	主备
zookeeper	4G	80G	2 核	集群

3. 产品安装

3.1. 环境要求

系统：RedHat6.9

JDK：Oracle Java Standard Edition 8 或者 OpenJDK8

防火墙：保证各节点对应端口可访问

授权：对应 shell 脚本有执行权限

3.2. 软件拷贝

- 1、选择一台服务器作为安装配置中心
- 2、将安装包拷贝到对应的服务器

3、该服务器可以访问所有节点

3.3. 安装配置

```
-rwxr-xr-x. 1 root root 370 3月 26 11:58 restart.sh
-rwxr-xr-x. 1 root root 370 3月 26 11:58 start.sh
-rwxr-xr-x. 1 root root 370 3月 26 11:58 stop.sh
-rwxr-xr-x. 1 root root 370 3月 26 11:58 clean.sh
drwxr-xr-x. 6 root root 4.0K 3月 26 11:56 conf
drwxr-xr-x. 2 root root 4.0K 3月 26 11:55 zookeeper
drwxr-xr-x. 18 root root 4.0K 3月 26 11:55 software
drwxr-xr-x. 5 root root 4.0K 3月 26 11:52 local
drwxr-xr-x. 2 root root 4.0K 3月 26 11:52 gcc
drwxr-xr-x. 2 root root 4.0K 3月 26 11:52 cassandra
drwxr-xr-x. 2 root root 4.0K 3月 26 11:52 gateway
drwxrwxr-x. 2 root root 4.0K 3月 26 11:52 gw_manager
drwxr-xr-x. 2 root root 4.0K 3月 26 11:52 jdk
drwxr-xr-x. 2 root root 4.0K 3月 26 11:52 redis
drwxr-xr-x. 2 root root 4.0K 3月 26 11:52 logs
drwxr-xr-x. 2 root root 4.0K 3月 26 11:52 MariaDB
-rwxr-xr-x. 1 root root 370 3月 26 11:52 install.sh
-rwxr-xr-x. 1 root root 146 3月 26 11:52 install_local.sh
```

install_local.sh 为安装中心配置脚本。

install.sh 为各节点安装脚本。

所有服务配置文件均在 conf 文件夹中。

clean.sh --清除系统中已经安装的软件。

start.sh --启动各个节点服务。

stop.sh --停止各个节点服务。

restart.sh --重启各个节点服务。

software 目录为所有软件包所在。

logs 为安装中产生的日志文件存放目录。

其他目录均为各个安装模块，每个模块对应安装一个软件。

具体每个软件安装的配置：

进入 conf 文件夹

cd conf/

```
-rw-r--r--. 1 root root 276 3月 26 11:52 bashrc.conf
-rw-r--r--. 1 root root 27 3月 26 11:52 cassandra_slave.conf
drwxrwxr-x. 2 root root 4096 3月 26 11:52 connector
drwxrwxr-x. 2 root root 4096 3月 26 11:52 gateway
-rw-r--r--. 1 root root 27 3月 26 11:52 gcc_slave.conf
drwxr-xr-x. 2 root root 4096 3月 26 11:52 gw_manager
-rw-r--r--. 1 root root 53 3月 26 11:52 jdk_slave.conf
drwxrwxr-x. 2 root root 4096 3月 26 11:52 mariadb
-rw-r--r--. 1 root root 189 3月 26 11:52 redis_cluster_slave.conf
-rw-r--r--. 1 root root 46564 3月 26 11:52 redis.conf
-rw-r--r--. 1 root root 53 3月 26 11:52 redis_slave.conf
-rw-r--r--. 1 root root 27 3月 26 11:52 zookeeper_slave.conf
```

带有

_slave.conf 后缀的文件为各个软件需要安装的服务节点设置文件，格式为：

服务器用户名，密码，ip 地址，均以逗号分隔，多个服务器则换行；

其中 mariadb_slave.conf 中存在第四列，为数据库 root 用户的初始密码；
其中 redis_slave.conf 为指定需要安装 redis 软件环境的配置文件，
redis_cluster_slave.conf 为具体每个节点服务位置文件，多了一个端口号，
安装过程中会根据这个文件来设定 redis 集群，例如：

```
root,password,10.204.127.8,6000  
root,password,10.204.127.68,6001  
root,password,10.204.127.68,6002  
root,password,10.204.127.68,6003  
root,password,10.204.127.8,6004  
root,password,10.204.127.8,6005
```

则程序会自动将 redis 前三个设定为 master 后面三个依次为前面三个 master 的 slave 节点。

3.4. 执行安装

各个服务对应的配置都完毕后便可执行安装程序；

首先在安装中心需要执行 ./install_local.sh

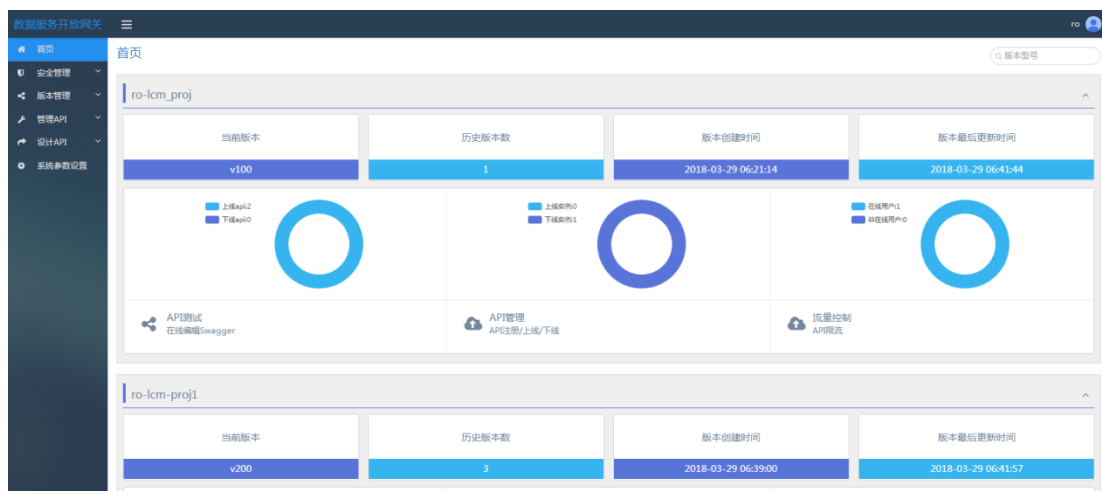
安装完毕后，便可执行 ./install.sh

执行过程中可以看到详细的安装和配置日志，执行完毕后所有服务便会启动。

4. 产品使用

4.1. 首页

首页展示了当前用户管理的项目服务 API 的总体情况，包括当前版本，历史版本，创建时间，更新时间以及版本上下线的数量概览统计等。



4.2. 安全管理

安全管理中包含了角色管理，项目管理，用户管理和授权管理。

4.2.1. 角色管理

角色管理可以创建用户角色，可以按照需要创建各种角色，角色可以分配系统菜单访问权限。

角色管理

系统管理 > 安全管理 > 角色管理

创建

1 填写基本信息 2 菜单权限分配 3 确认填写信息 4 完成

角色名: admin

角色描述: ro-lcm ro_role

总数: 3

下一步 取消

4.2.2. 项目管理

项目管理创建项目名称，项目你可以这样理解，项目即一组 API 服务，对外发布的服务 API 集合，可能对应很多不同的项目，一组 API 集合提供给 A 方使用，另一组 API 集合提供给 B 方使用，这时，就可以创建项目 A 和项目 B，方便后续管理。

项目管理

系统管理 > 安全管理 > 项目管理

创建

1 填写基本信息 2 确认填写信息 3 完成

项目名称: ro-lcm-proj1

项目描述: ro-lcm_proj

总数: 2

下一步 取消

4.2.3. 用户管理

用户管理可以根据需要创建新用户，创建用户过程中，需要绑定之前创建的角色和项目，角色和项目都是支持多选的，选择的角色决定了创建的用户登录后的菜单访问权限，选择的项目决定了版本管理和 API 管理中的可见项目。

4.2.4. 授权管理

授权管理包括凭证创建、凭证续期、根据凭证进行 API 服务限流。凭证管理粒度可细化到用户、用户下面的项目、单个 API 三个维度（默认只支持到用户，即该用户创建的凭证可以访问其管辖项目下所有的 API）。凭证可以是永久凭证或有期限的凭证，凭证到期后可续期。

在授权管理中可以在这里设置一个或多个服务的 ID，每个授权信息包含服务有效期，流控的默认阈值，创建完成后，您会得到一个对应 ID 的密钥，用户用这个 ID 和密钥可以动态的获取 token，通过此 token 来获取数据服务 API 的访问权限。在接口说明章节中有详细的接口使用方法。

4.3. 版本管理

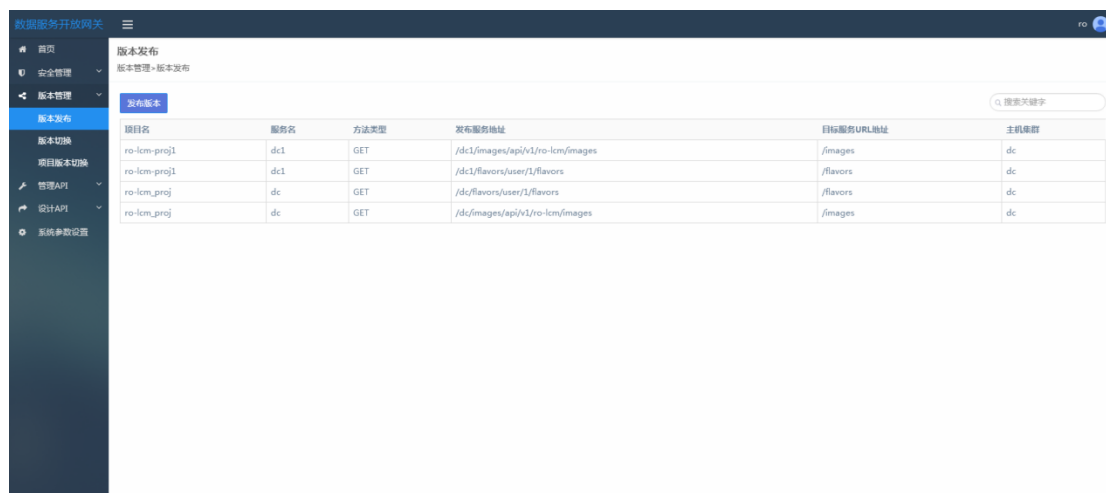
版本管理对上线的 API 服务进行版本控制，包含版本发布、升级、回退。可以按项目或者单个 API 进行操作。支持多版本同时在线，支持灰度发布。版本切换回退不需要重启平台，相关系统业务不感知。

版本管理的作用是发布和切换各个不同项目版本，使之能对外提供服务能力。为

什么要有版本发布呢，因为你在项目下增加了新的 API 服务，或者对原有的 API 服务 URL 或方法进行了修改，为了使修改生效，您可以基于项目来批量发布 API 服务，而不需要一个个手工点击上线动作。发布后的版本立即生效，可以在项目版本切换中看到当前使用的版本号和接口列表（后面会有介绍）。

4.3.1. 版本发布

下图列出了当前在线的项目 API 服务列表。



项目名称	服务名	方法类型	发布服务地址	目标服务URL地址	主机集群
ro-lcm-proj1	dc1	GET	/dc1/images/api/v1/ro-lcm/images	/images	dc
ro-lcm-proj1	dc1	GET	/dc1/flavors/user/1/flavors	/flavors	dc
ro-lcm_proj	dc	GET	/dc/flavors/user/1/flavors	/flavors	dc
ro-lcm_proj	dc	GET	/dc/images/api/v1/ro-lcm/images	/images	dc

点击发布版本



版本发布

版本发布

版本信息:

版本标识

版本描述

选择项目:

ro-lcm_proj

×

Q

创建

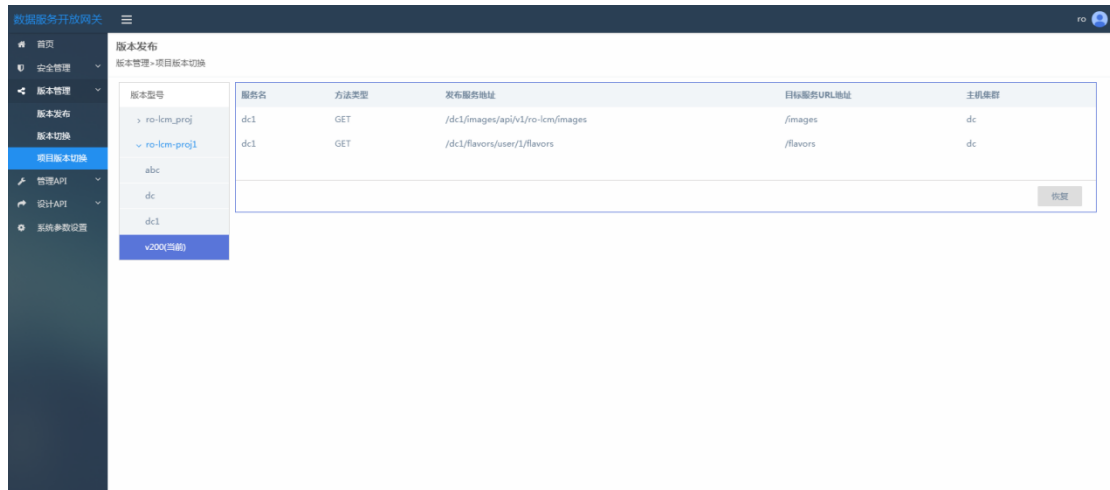
关闭

项目名称	服务名
ro-lcm-proj1	dc1
ro-lcm-proj1	dc1
ro-lcm_proj	dc
ro-lcm_proj	dc

版本信息和描述，根据您的规则填写，点击选择项目会列出当前用户下的所有项目，你也可以一次选择多个项目一起发布。

4.3.2. 版本切换

项目版本切换，当需要版本回退或回归测试时，可以随时切换到想要的服务版本，选择版本后，点击切换，即可切换到你选择的版本。此时新版本的 API 服务提供对外服务能力，原版本自动下线。



4.4. 管理 API

4.4.1. 集群管理

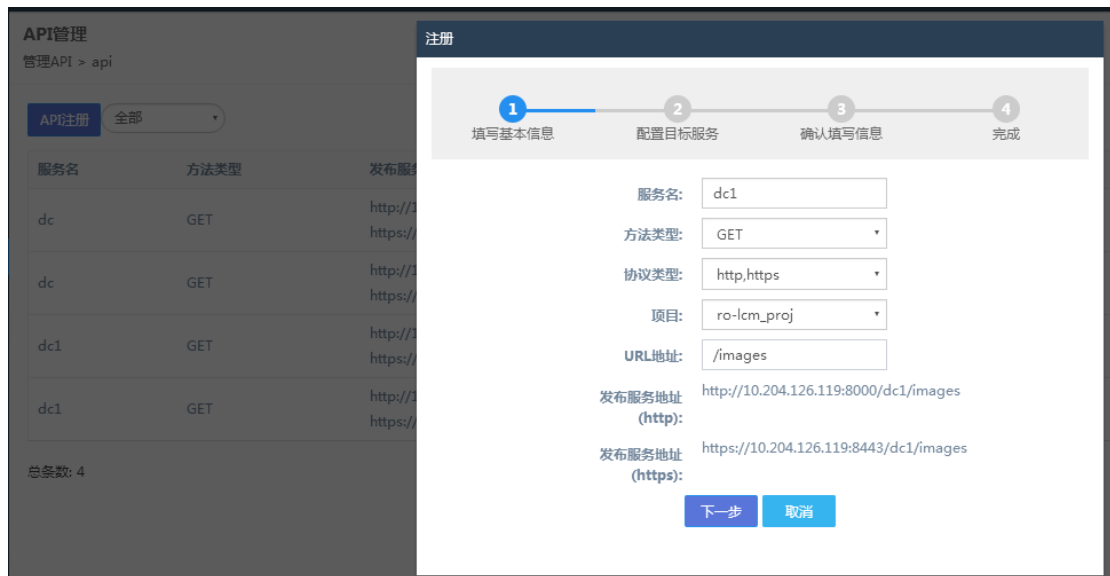
当对外提供的服务是多实例部署时，可以在界面通过创建多个实例，为每个实例配置 IP 和服务端口号，同时配置访问分配权重，数据服务网关会根据配置的权重配置分发请求。当然如果服务本身已经配置了负载均衡或只是单实例部署，则配置一条 IP 和端口即可。这里配置的集群在下面的 API 管理配置是路由目的地。



4.4.2. API 管理

API 管理是提供 API 注册、路由、上线、下线等功能的主要配置入口，如下图，点击 API 注册：

第一步，填写基本信息，服务名是发布 API 接口的第一级后缀，自定义，可重复；方法类型有 GET, POST, PUT, DELETE 可以选择，取决于需要发布成什么接口；项目在下拉框中选择之前定义好的项目，选择某个项目后，本次新增的 API 接口归属于这个项目树下；URL 地址以/开头，填写需要发布的 API 地址；



The screenshot shows the 'API管理' (API Management) interface with the 'API注册' (API Registration) tab selected. On the left, a table lists API services. On the right, the '注册' (Registration) wizard is in step 1, '填写基本信息' (Fill in basic information).

服务名	方法类型	发布服务地址
dc	GET	http://10.204.126.119:8000/dc/
dc	GET	https://10.204.126.119:8443/dc/
dc1	GET	http://10.204.126.119:8000/dc1/
dc1	GET	https://10.204.126.119:8443/dc1/

总条数: 4

注册

1 填写基本信息 2 配置目标服务 3 确认填写信息 4 完成

服务名: dc1
方法类型: GET
协议类型: http,https
项目: ro-lcm_proj
URL地址: /images
发布服务地址 (http): http://10.204.126.119:8000/dc1/images
发布服务地址 (https): https://10.204.126.119:8443/dc1/images
下一步 取消

第二步，点击下一步进入配置目标服务，目标服务 URL 地址填写对外提供 API 服务的原生 URL，主机集群选择在集群管理中配置的集群；



The screenshot shows the '注册' (Registration) wizard in step 2, '配置目标服务' (Configure target service).

1 填写基本信息 2 配置目标服务 3 确认填写信息 4 完成

目标服务URL地址: /api/v1/ro-lcm/images
主机集群: dc
上一步 下一步 取消

点击下一步进入信息确认界面，即可完成配置。

















The screenshot shows the '注册' (Registration) wizard in step 3, '确认填写信息' (Confirm fill information).

1 填写基本信息 2 配置目标服务 3 确认填写信息 4 完成

服务名: dc1
方法类型: GET
发布服务地址(http): http://10.204.126.119:8000/dc1/images
发布服务地址(https): https://10.204.126.119:8443/dc1/images
目标服务URL地址: /api/v1/ro-lcm/images
主机集群: dc
上一步 下一步 取消



在 API 管理的列表中可以看到刚刚配置完成的一条 API，如下图：

<div>API注册 全部</div>					
服务名	方法类型	发布服务地址	目标服务URL地址	主机集群	操作
dc	GET	http://10.204.126.119:8000/dc/flavors https://10.204.126.119:8443/dc/flavors	/user/1/flavors	dc	   
dc	GET	http://10.204.126.119:8000/dc/images https://10.204.126.119:8443/dc/images	/api/v1/ro-lcm/images	dc	   
dc1	GET	http://10.204.126.119:8000/dc1/flavors https://10.204.126.119:8443/dc1/flavors	/user/1/flavors	dc	   
dc1	GET	http://10.204.126.119:8000/dc1/images https://10.204.126.119:8443/dc1/images	/api/v1/ro-lcm/images	dc	   

总条数: 4

原服务对外提供的地址是 10.204.126.130 端口号 50000，完整的 URL 是 http://10.204.126.130:50000/api/v1/ro-lcm/images，经过 API 注册后，服务完整 URL 变成了 http://10.204.126.119:8000/dc1/images，同时也可以通过 https 访问，完成了 URL 的转换。此时使用此 URL 对外部提供服务时，具备了自动负载均衡，限流的能力。

4.4.3. 流量控制

流量控制可以在默认的项目流量控制策略之外，针对特定的 API 设置自己的流量控制策略，对上线使用的 API 服务支持流量门限设定控制，达到设定条件时可以根据策略主动限流；主要的 API 服务存在不可用风险的时候，根据设定可以对登记注册等级较低的应用进行限流，优先保证高等级服务的可用性。最小粒度为秒，一般按照秒，分，小时，天，月，年数字逐渐递增，以最粗粒度的时间限制值为准。

The screenshot shows a web interface for traffic control. On the left, there's a sidebar with '流量控制' (Traffic Control) and '管理API > 流量控制'. A button '增加流量控制' (Add Traffic Control) is visible. Below it, a table header shows 'API服务名' and '次/秒', and a status '总条数: 0'. The main area is titled '新增流量控制' (Add Traffic Control). It contains a form with the following fields: 'API:' (a dropdown menu showing '/dc1/flavors'), '秒:' (seconds), '分:' (minutes), '小时:' (hours), '天:' (days), '月:' (months), and '年:' (years). At the bottom of the form are three buttons: '保存' (Save), '取消' (Cancel), and '完成' (Finish).

4.5. 设计 API

支持用户完成开发的接口在平台上进行 REST-API 格式的校验验证；支持用户完成开发的接口在平台上进行 REST-API 格式的校验验证；支持用户完成开发的接口实现在平台上做 Mock 测试，模拟调试，以软件最终发布功能为准。

4.6. 系统参数设置

系统参数中的参数项由系统默认提供，管理员可以修改和设置实际生效值，可配置的项为 api_token 有效期(秒)、api 网关服务地址(http)、api 网关服务地址(https)、api 网关管理地址等。

5. 接口说明

访问者需要获取 token，才能访问提供给他的数据服务接口。

5.1. 获取 token 接口

URL:

http://部署数据网关服务的机器 IP:18011/access_token

接口类型:

POST

Body 参数:

```
{"secretkey":"ZDc0NjI5ZjgtNTYxNC00MzhkLTkwMjctNzA1Nzg4","appid":"ro-au"}
```

其中 secretkey 是在安全管理的授权管理中创建的应用 ID 对应的密钥，appid 是应用 id 的名称，如下图：



返回值：

```
{
  "obj": "fc64d774f48f42ce888af77132d6f008",
  "result_code": 200
}
```

Obj 参数的内容即本次访问需要的 token，token 的有效期和时限在系统参数设置中配置。

5.2. 使用 token 方法

获取到了 token，调用者需要在代码的请求消息头中加入 open_api_token 使其生效。

以 curl 测试举例：

源 API 服务访问结果，留存与服务网关提供的接口结果做对比

```
[root@localhost ~]# curl http://10.204.126.130:5000/api/v1/ro-lcm/images
[
  {
    "checksum": "1472472796829",
    "container_format": "bare",
    "create_user": "1",
    "createdAt": "2017-10-21 14:45:07",
    "disk_format": "qcow2",
    "id": "2bfd0c81fd52370",
    "min_disk": 10,
    "min_ram": 1,
    "name": "st",
    "os_type": "ubuntu",
    "os_version": "14.04",
    "owner": null,
    "protected": false,
    "self": "",
    "size": 11075584,
    "status": "success",
    "updatedAt": "None",
    "vdc_id": "1",
    "vdc_name": "YK_DC",
    "visibility": true
  }
]
```

访问请求中不加 token 访问, 访问数据服务网关提供的接口无法返回结果

```
[root@localhost ~]# curl http://10.204.126.119:8000/dc/images
{"message":"The request occur error!open_api_token not exists"}
[root@localhost ~]#
```

加上消息头后, 访问数据服务网关提供的接口, 结果同源接口。

```
[root@localhost ~]# curl -H 'open_api_token:13dc7e330817799ff2262047fb4f7a2b' http://10.204.126.119:8000/dc/images
[
  {
    "checksum": "1472472796829",
    "container_format": "bare",
    "create_user": "1",
    "createdAt": "2017-10-21 14:45:07",
    "disk_format": "qcow2",
    "id": "2bfd0c81fd52370",
    "min_disk": 10,
    "min_ram": 1,
    "name": "st",
    "os_type": "ubuntu",
    "os_version": "14.04",
    "owner": null,
    "protected": false,
    "self": "",
    "size": 11075584,
    "status": "success",
    "updatedAt": "None",
    "vdc_id": "1",
    "vdc_name": "YK_DC",
    "visibility": true
  }
]
```