|  |  |
| --- | --- |
| 文档编号 | IAAS-DDS 002 |
| 文档版本 | 1.0 |
| 拟制人 | 段彬彬 |
| 日期 | 2016.5.31 |

**EayunCloud公有云业务管理平台**

**（对象存储CDN加速服务）**

**概要设计说明书**

变更说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **变更位置** | **变更说明** | **作者** |
| 2016.05.31 | 1.0 |  | 新建 | 段彬彬 |
| 2016.06.01 | 1.0 |  | 补充接口描述 | 段彬彬 |
| 2016.06.15 | 1.0 |  | 补充CDN接口封装及提供给底层的object和bucket变更通知接口流程图 | 张帆 |
| 2016.06.16 | 1.0 |  | 调整补充Service方法 | 段彬彬 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[变更说明 2](#_Toc454785820)

[目录 3](#_Toc454785821)

[1.1 业务分析 6](#_Toc454785822)

[1.1.1 概述 6](#_Toc454785823)

[1.1.2 域名状态分析 6](#_Toc454785824)

[1.1.3 CDN加速对原系统的影响 7](#_Toc454785825)

[1.1.3.1 Bucket模块需修改或替换的方法 7](#_Toc454785826)

[1.1.4 用例图 7](#_Toc454785827)

[1.1.4.1 Bucket基础操作 7](#_Toc454785828)

[1.1.4.2 CDN基础加速 7](#_Toc454785829)

[1.1.4.3 CDN与防盗链服务对接 8](#_Toc454785830)

[1.1.4.4 CDN加速计费 8](#_Toc454785831)

[1.1.4.5 文件刷新 8](#_Toc454785832)

[1.1.5 流程图 9](#_Toc454785833)

[1.1.5.1 Bucket基础操作 9](#_Toc454785834)

[1.1.5.1.1 查询列表 9](#_Toc454785835)

[1.1.5.1.2 创建bucket 9](#_Toc454785836)

[1.1.5.2 CDN基础加速 9](#_Toc454785837)

[1.1.5.2.1 开启CDN加速 9](#_Toc454785838)

[1.1.5.2.2 关闭CDN加速 11](#_Toc454785839)

[1.1.5.2.3 获取文件URL 12](#_Toc454785840)

[1.1.5.2.4 访问文件 12](#_Toc454785841)

[1.1.5.2.5 删除bucket 13](#_Toc454785842)

[1.1.5.3 CDN与防盗链对接 13](#_Toc454785843)

[1.1.5.4 CDN加速流量计费 13](#_Toc454785844)

[1.1.5.4.1 需增加计划任务 13](#_Toc454785845)

[1.1.5.4.1.1 每小时计划任务（obs.cdn.1h）： 13](#_Toc454785846)

[1.1.5.4.1.2 每天计划任务（obs.cdn.1d）： 14](#_Toc454785847)

[1.1.5.4.1.3 每周计划任务（obs:cdnSortByDownload）： 15](#_Toc454785848)

[1.1.5.4.1.4 每月计划任务（obs.cdn.1mon）： 16](#_Toc454785849)

[1.1.5.4.2 CDN下载流量显示 16](#_Toc454785850)

[1.1.5.5 文件刷新 17](#_Toc454785851)

[1.1.5.5.1 底层调用接口 17](#_Toc454785852)

[1.1.5.5.2 计划任务：处理底层调用通知 22](#_Toc454785853)

[1.1.6 中间状态处理（若使用UpYun不需要此模块） 23](#_Toc454785854)

[1.1.7 记录API相关日志 25](#_Toc454785855)

[1.1.7.1 CDN日志 25](#_Toc454785856)

[1.1.7.2 DNS日志 25](#_Toc454785857)

[1.1.8 UpYun API 25](#_Toc454785858)

[1.1.8.1 创建 25](#_Toc454785859)

[1.1.8.2 启用 26](#_Toc454785860)

[1.1.8.3 禁用 26](#_Toc454785861)

[1.1.8.4 查询流量 27](#_Toc454785862)

[1.1.8.5 删除（bucket） 27](#_Toc454785863)

[1.1.8.6 缓存刷新 27](#_Toc454785864)

[1.1.9 DNS API 27](#_Toc454785865)

[1.1.9.1 添加记录 28](#_Toc454785866)

[1.1.9.2 设置记录状态 29](#_Toc454785867)

[1.1.9.3 获取记录信息 29](#_Toc454785868)

[1.1.9.4 删除记录 31](#_Toc454785869)

[1.1.10 本次新增的消息队列和计划任务 31](#_Toc454785870)

[1.1.10.1 消息队列 31](#_Toc454785871)

[1.1.10.2 计划任务 32](#_Toc454785872)

[1.2 Model设计 32](#_Toc454785873)

[1.2.1 CdnBucket 32](#_Toc454785874)

[1.2.2 BaseCdnBucket 33](#_Toc454785875)

[1.2.3 cdn.api.log 33](#_Toc454785876)

[1.2.4 dns.api.log 33](#_Toc454785877)

[1.2.5 ObsUsedType 34](#_Toc454785878)

[1.2.6 BucketStorageBean 34](#_Toc454785879)

[1.2.7 ObsBucket 34](#_Toc454785880)

[1.3 Controller 34](#_Toc454785881)

[1.3.1 Object和Bucket操作变更通知接口 34](#_Toc454785882)

[1.3.1.1 包命、类名 34](#_Toc454785883)

[1.3.1.2 请求地址和伪代码 34](#_Toc454785884)

[1.3.2 ObsBucketController 36](#_Toc454785885)

[1.3.2.1 getBucketPageList(getBucketPageList)OK 36](#_Toc454785886)

[1.3.3 ObsCdnBucketController（新增） 37](#_Toc454785887)

[1.3.3.1 openCDN(opencdn) 37](#_Toc454785888)

[1.3.3.2 closeCDN(closecdn) 38](#_Toc454785889)

[1.3.3.3 getCdnFlowByBucket(getcdnflowbybucket) 38](#_Toc454785890)

[1.3.4 ObsUsedController 39](#_Toc454785891)

[1.3.4.1 getObsResources(getObsResources)OK 39](#_Toc454785892)

[1.4 Service接口 39](#_Toc454785893)

[1.4.1 CDN接口封装 40](#_Toc454785894)

[1.4.2 ObsBucketService 41](#_Toc454785895)

[1.4.2.1 getBucketPage 41](#_Toc454785896)

[1.4.3 ObsUsedService 42](#_Toc454785897)

[1.4.3.1 getObsUsedList 42](#_Toc454785898)

[1.4.4 obsBaseStorageService 42](#_Toc454785899)

[1.4.4.1 getUrl 42](#_Toc454785900)

[1.4.5 ObsCdnBucketService（新增） 42](#_Toc454785901)

[1.4.5.1 enableDomain 42](#_Toc454785902)

[1.4.5.2 disableDomain 42](#_Toc454785903)

[1.4.5.3 getCDNFlowData 43](#_Toc454785904)

[1.4.5.4 getUnDeleteListByCusId 43](#_Toc454785905)

[1.4.5.5 getUnDeleteByCusAndName 43](#_Toc454785906)

[1.4.5.6 addCdnLog（操作CDN时调用） 44](#_Toc454785907)

[1.4.5.7 addDnsLog（操作DNS时调用） 44](#_Toc454785908)

[1.5 增加计划任务 44](#_Toc454785909)

[1.5.1 CdnDetailGatherThread 44](#_Toc454785910)

[1.5.2 CdnDaySummaryThread 45](#_Toc454785911)

[1.5.3 CdnWeekSummaryThread 45](#_Toc454785912)

[1.5.4 CdnMonthSummaryThread 45](#_Toc454785913)

[1.5.5 CdnRefreshObjThread 45](#_Toc454785914)

[1.5.6 CdnRefreshBucketThread 45](#_Toc454785915)

## 业务分析

### 概述

易云对象存储提供针对bucket的CDN加速服务，bucket开启加速服务后，默认获取的object的地址均为CDN加速后的地址。根据实际需要，cdn加速后的地址不能直接暴露给用户，因此需要通过DNS供应商将cdn加速后的地址绑定到我们提供的地址上，展示给用户使用。

用户设置防盗链功能后需要与CDN功能进行对接，将白名单服务同步到CDN供应商处。

用户开启CDN加速功能后，以下过程产生的流量将对用户进行计费：

①、用户从cache节点获取数据时候的CDN下载流量，由CDN供应商接口获取。

②、cache节点从对象存储服务获取源数据的回源流量，由CDN供应商接口获取。

CDN下载流量为新增的计费项目，由；回源流量为对象存储服务的普通下载流量。1.1版本所有需要统计的流量数据包括：

①、流入流量（上传流量）

②、流出流量（下载流量）

③、CDN下载流量

④、CDN回源流量（暂无法分离）

⑤、内网流入流量（暂无）

⑥、内网流出流量（暂无）

新版ECSC需体现CDN加速流量，规则展现与“下载流量”一致，两者各自展示，无联系。

用户上传同名文件或删除该文件时，调用CDN供应商的刷新功能，将cache节点内的该文件清空。

### 域名状态分析

**此文档主要分析了网宿和****UpYun两家供应商的API，并尽可能将其封装起来**。

网宿CDN域名状态分析：

经过实际验证，在做下面的操作时，加速域名的状态改变如下：

CDN加速域名状态：<status>{Deployed or InProgress}</status>  
 Deployed（已完成）；InProgress（创建中）

是否激活：<enabled>{true or false}</enabled>

加速域名创建过程：

①、enabled设为true

②、status置为InProgress—>Deployed

禁用过程：

①、enabled由初始状态为true，设为false

②、status变化为：**Deployed —>InProgress—>Deployed**

启用过程：

①、enabled由初始状态为false，设为true

②、status变化为：**Deployed —>InProgress—>Deployed**

删除过程：

①、清除该加速域名的所有信息

以上过程均为异步操作，测试中过程大约持续6分钟

UpYun供应商CDN域名状态分析：

UpYun所有操作均为同步任务，操作成功返回，则认为状态设置完成。

### CDN加速对原系统的影响

#### Bucket模块需修改或替换的方法

①、bucket列表查询：沿用原来的方法，只是增加一个CDN状态字段，从关联表里面查出来。

②、删除bucket不需要更改，对于CDN和DNS的后续处理在得到底层的通知后再进行。

### 用例图

#### Bucket基础操作



#### CDN基础加速



#### CDN与防盗链服务对接



#### CDN加速计费



#### 文件刷新

底层做增、删、改object的操作，对bucket做删除操作时，通知到系统，系统调用CDN接口刷新文件：





### 流程图

#### Bucket基础操作

##### 查询列表



##### 创建bucket

仍沿用之前的方法



#### CDN基础加速

##### 开启CDN加速



##### 关闭CDN加速



##### 获取文件URL



##### 访问文件



##### 删除bucket

用户删除bucket时仍按照原先的操作，只处理底层相关信息；底层删除bucket时，调用系统接口通知，然后再根据底层反馈的信息做关于CDN的进一步处理。



#### CDN与防盗链对接

#### CDN加速流量计费

CDN加速需计费流量：CDN下载流量和CDN回源流量（无回源流量接口，可能已包含在下载流量中，回源流量理应只计一次费用）。

由于可能存在不同的提供商，因此**假设存在用户切换提供商的场景时，实际操作暂定为：首先将原先存在的CDN加速服务关闭掉，然后开启新的CDN加速服务，关联表里面原纪录不删除**，**CDN下载流量按照不同的供应商加以区分，日、周、月流量汇总也按照不同提供商加以区分，若前台不区分，将后台各个提供商的流量数据相加即可。**

##### 需增加计划任务

以下增加的计划任务均只为统计CDN加速产生的下载流量，普通的下载流量不计在内。

###### 每小时计划任务（obs.cdn.1h）：

每整点05分执行，bucket粒度，统计每个加速域名（bucket）的上一个整小时的流量汇总，如11时-12时的数据。

需要考虑CDN供应商的访问限制，并发调用风险。

obs.cdn.1h：{

"timestamp" :12:00:00 //日期（此时表示记录的数据为11时-12时之间的总流量）

"bucket\_name " : "", //bucket名称

"cus\_id" : "", //客户id

"domain\_id" : "", //bucket加速域名id(UpYun为生成bucket\_name)

"flow\_data" : "", //下载流量

"counter\_unit" : "B", //单位

"cdnProvider":"ChinaNetCenter" //**CDN提供商enum**<UpYun,ChinaNetCenter>

}



###### 每天计划任务（obs.cdn.1d）：

每天零时07分0秒执行，统计每个客户所有加速域名上一天整天的流量汇总，如5-23日的数据。查询obs.cdn.1h里的数据，按照客户信息、CDN提供商聚合。因为有不同的提供商，因此**每天的汇总流量数据可能不止一条**。

obs.cdn.1d：{

"timestamp" :2016-05-23 00:00:00//日期（此时表示记录的数据为5-23日整天的总流量）

"cus\_id" : "", //客户id

"flow\_data" : "", //下载流量

"counter\_unit" : "", //单位

"cdnProvider":"ChinaNetCenter" //**CDN提供商enum**<UpYun,ChinaNetCenter>

}



###### 每周计划任务（obs:cdnSortByDownload）：

obs:cdnSortByDownload{

"bucket\_name " : "", //bucket名称

"cus\_id" : "", //客户id

"domain\_id" : "", //bucket加速域名id

"flow\_data" : "", //下载流量（根据流量排序）

"counter\_unit" : "", //单位

}

每周一零时7分0秒执行，bucket粒度，统计每个加速域名（bucket）的上周的流量汇总排序，如05-16（上周一）至05-22（上周日）七天内的数据总和。

查询obs.cdn.1h里的数据，使用聚合，目前为所有CDN下载流量，不区分供应商。



###### 每月计划任务（obs.cdn.1mon）：

每月1日零时11分0秒执行，统计每个客户所有加速域名上一个自然月的流量汇总，如4月份的客户流量汇总。因为有不同的提供商，因此**每个客户一月的汇总流量数据可能不止一条。**

查询obs.cdn.1d里的数据，按照客户信息、CDN提供商使用聚合。

obs.cdn.1mon：{

"timestamp" :2016-04-01 00:00:00//日期（此时表示记录的数据为4月整月的总流量）

"cus\_id" : "", //客户id

"flow\_data" : "", //下载流量

"counter\_unit" : "", //单位

"cdnProvider":"ChinaNetCenter" //**CDN提供商enum**<UpYun,ChinaNetCenter>

}



##### CDN下载流量显示



资源报表中显示CDN下载流量



bucket详情中显示CDN下载流量曲线图表

#### 文件刷新

①、底层做增、删、改操作，调用接口将操作信息放入消息队列中。

②、系统开启计划任务，间隔1分钟，从消息队列中获取文件刷新记录，调用CDN接口刷新文件，上传同名文件或删除文件时，需要清除cache节点中的缓存文件。

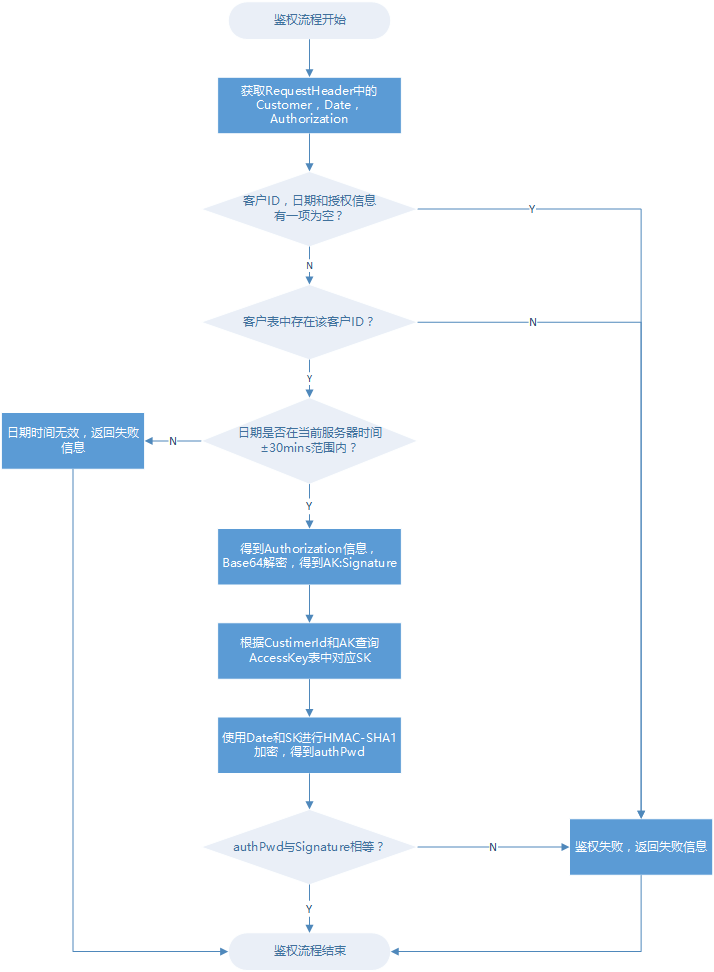
**底层调用情况：**

1. 、底层定时隔一段时间（如每5s）调用一次提供的接口，批量传入所操作的文件。

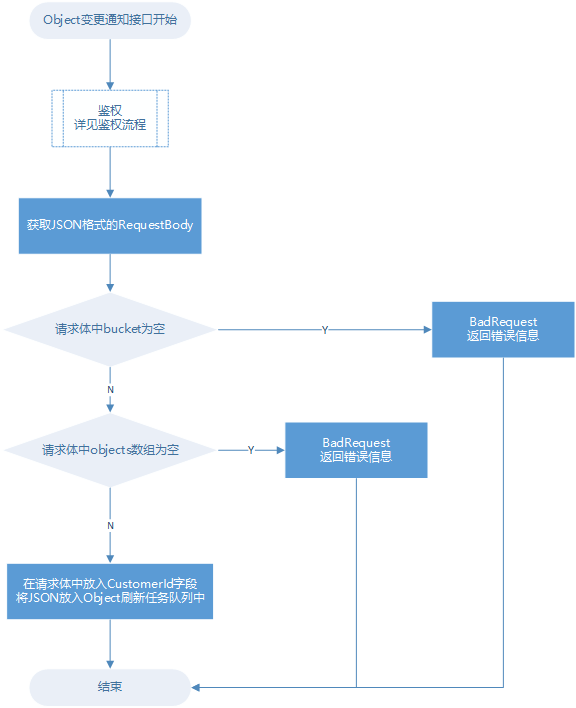
②、底层一次调用，对同一个文件如做了多次相同或不同的操作，只传入最新的一次操作，其余不传入。

##### 底层调用接口

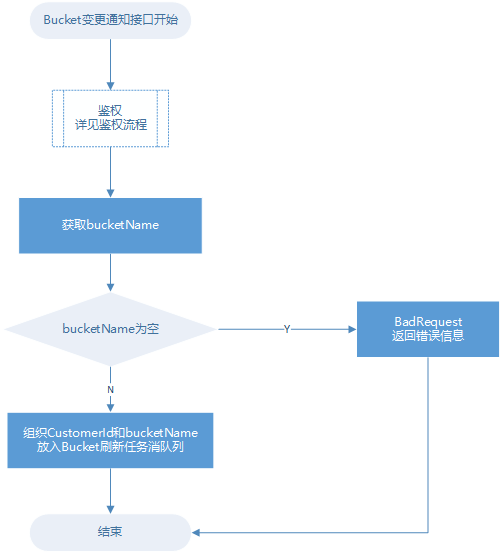
鉴权流程：



Object操作变更接口流程：



Bucket变更通知接口流程图：



底层调用系统接口，将操作object的信息放入消息队列和mongo日志中，消息队列用来调用CDN文件刷新接口依据，mongo日志作为底层调用日志存储。

底层删除bucket，将删除bucket的名称传入消息队列中。

消息队列：

**CDN\_REFRESH:SYNCOBJECT**{

customerId： //客户id

bucket: //bucket名称

objects:[

{

type: //文件操作类型（delete/upload/update）

object: //文件相对路径

}

{

type: //文件操作类型（delete/upload/update）

object: //文件相对路径

}

......

]

}

**CDN\_REFRESH:SYNCBUCKET**{

customerId： //客户id

bucket: //bucket名称

}

mongo日志：

cdn.refresh.object {

timestamp: //日期

customerId： //客户id

bucket: //bucket名称

objects:[

{

type: //文件操作类型（delete/upload/update）

object: //文件相对路径

}

......

]

}

cdn.refresh.bucket {

timestamp: //日期

customerId： //客户id

bucket: //bucket名称

}



##### 计划任务：处理底层调用通知

包括处理操作object和操作bucket处理。

增加两个每分钟的计划任务，每分钟0秒执行，取出**CDN\_REFRESH:SYNCOBJECT、CDN\_REFRESH:SYNCBUCKET**消息队列里的信息，调用CDN的文件刷新接口（目前即清除文件缓存接口）处理，或禁用加速域名（删除bucket时）。





### 中间状态处理（若使用UpYun不需要此模块）

**此种情况只存在于使用网宿API**时，由于操作异步，执行**创建、启用、禁用**操作后，加速域名配置状态不知何时完成，需要等待时间再次确定执行结果的状态。参考云资源同步使用两步解决；

①、使用消息队列，增加每1分钟执行一次的计划任务，同步bucket信息，执行状态更新操作。

②、若查询列表中存在中间状态(cdn\_status='1')的bucket信息，则每1分钟（这个时间可以根据实际需要调整）查询一次加速域名状态，**这个操作应尽量不涉及再次查询底层**，只改变状态。

**CDN\_RESOURCE:SYNCCDN**{

requestId //执行操作任务号

bucketId //数据库bucket的id

bucketName //bucket名称

domainId //CDN中加速域名id

count: //只记录次数，不限制最多执行次数

type: //操作类型（创建域名create，禁用域名disable，启用域名enable）

}



### 记录API相关日志

#### CDN日志

记录和CDN有关的操作信息：

cdnProvider： //CDN提供商

UpYun

ChinaNetCenter

operationCdn：

//ChinaNetCenter

......

//UpYun

创建空间（CREATE\_BUCKETS）

配置回源（CONFIG\_SOURCE）

配置缓存规则（CONFIG\_CACHE）

绑定加速域名（BIND\_DOMAIN）

设置空间外链状态（*CONFIG\_VISIBLE*）

查询流量（QUERY\_BANDWIDTH）

刷新缓存（**REFRESH**\_CACHE）

#### DNS日志

记录和CDN有关的操作信息：

operationDns：

创建记录（CREATE\_RECORD）

设置记录状态（CONFIG\_RECORD）

删除记录（DELETE\_RECORD）

### UpYun API

UpYun所有操作均为同步任务

#### 创建

UpYun创建过程分为几个步骤：

①、创建空间（/buckets/）

|  |  |  |  |
| --- | --- | --- | --- |
|  | **必选** | **数据类型及范围** | **说明** |
| bucket\_name | true | String | 服务名称，只能包含小写字母、数字和中划线,20位以内（含20位） |
| type | true | String | 服务类型(file/ucdn) |
| quota | false | Integer | 服务存储配额，设置为0或者流空表示无限制 |

bucket\_name：eayun系统为每个bucket生成一个唯一的标示，用于与UpYun API交互的依据。

加速域名地址为：{bucket\_name }.file.eayun.com

②、配置回源（https://api.upyun.com/v2/buckets/cdn/source/）

|  |  |  |  |
| --- | --- | --- | --- |
|  | **必选** | **数据类型及范围** | **说明** |
| bucket\_name | true | String | 空间名 |
| domain | false | String | 回源请求域名；（加速地址） |
| cdn | true | json | cdn 详细设置信息 |
| source\_type | true | enum('http','https', 'protocol\_follow') | 回源方式设置：http：http 协议回源；https：https 协议回源；protocol\_follow：协议跟随，根据请求的实际协议类型进行回源 |
| domain\_follow | true | enum('enable','disable') | 回源是否开启域名跟随 |

当开启 domain\_follow 时(domain\_follow=enable)，不需要传递 domain 字段，否则需要 domain 。

③、配置缓存规则（/v2/buckets/cdn/cache/）

|  |  |  |  |
| --- | --- | --- | --- |
|  | **必选** | **数据类型及范围** | **说明** |
| bucket\_name | true | String | 空间名 |
| specific\_rules | true | json | 特殊缓存规则 |
| nocache\_list | true | json | 不缓存列表 |

④、绑定加速域名（/buckets/domains/）

|  |  |  |  |
| --- | --- | --- | --- |
|  | **必选** | **数据类型及范围** | **说明** |
| bucket\_name | true | String | 空间名 |
| domain | true | String | 长度小于 60 位字符的英文域名 |

#### 启用

绑定加速域名（/buckets/domains/）

|  |  |  |  |
| --- | --- | --- | --- |
|  | **必选** | **数据类型及范围** | **说明** |
| bucket\_name | true | String | 空间名 |
| domain | true | String | 长度小于 60 位字符的英文域名 |

#### 禁用

删除绑定域名（/buckets/domains/）

|  |  |  |  |
| --- | --- | --- | --- |
|  | **必选** | **数据类型及范围** | **说明** |
| bucket\_name | true | String | 空间名 |
| domain | true | String | 域名 |

#### 查询流量

（https://api.upyun.com/v2/statistics/）

|  |  |  |  |
| --- | --- | --- | --- |
|  | **必选** | **数据类型及范围** | **说明** |
| bucket\_name | false | String | 空间名 |
| start\_time | false | String | 开始时间 |
| end\_time | false | String | 结束时间 |
| domain | false | String | 域名 |

#### 删除（bucket）

删除绑定域名（/buckets/domains/）

|  |  |  |  |
| --- | --- | --- | --- |
|  | **必选** | **数据类型及范围** | **说明** |
| bucket\_name | true | String | 空间名 |
| Domain | true | String | 域名 |

#### 缓存刷新

（https://api.upyun.com/purge/）

|  |  |  |  |
| --- | --- | --- | --- |
|  | **必选** | **数据类型及范围** | **说明** |
| urls | true | String | 待刷url |

### DNS API

公共参数：

* login\_token 用于鉴权的 API Token
* format {json,xml} 返回的数据格式，可选，默认为xml，建议用json
* lang {en,cn} 返回的错误语言，可选，默认为en，建议用cn
* error\_on\_empty {yes,no} 没有数据时是否返回错误，可选，默认为yes，建议用no
* user\_id 用户的ID，可选，仅代理接口需要，用户接口不需要提交此参数

共通返回：

* -1 登陆失败
* -2 API使用超出限制
* -3 不是合法代理 (仅用于代理接口)
* -4 不在代理名下 (仅用于代理接口)
* -7 无权使用此接口
* -8 登录失败次数过多，帐号被暂时封禁
* 85 帐号异地登录，请求被拒绝
* -99 此功能暂停开放，请稍候重试
* 1 操作成功
* 2 只允许POST方法
* 3 未知错误
* 6 用户ID错误 (仅用于代理接口)
* 7 用户不在您名下 (仅用于代理接口)
* 83 该帐户已经被锁定，无法进行任何操作
* 85 该帐户开启了登录区域保护，当前IP不在允许的区域内

#### 添加记录

用于第一次开通CDN

接口地址：https://dnsapi.cn/Record.Create

HTTP请求方式：POST

请求参数：

公共参数

domain\_id 域名, 必选（如file.eayun.com）

sub\_domain 主机记录, 如 www, 默认@，可选

record\_type 记录类型，通过API记录类型获得，大写英文，'CNAME', 必选

record\_line 记录线路，通过API记录线路获得，中文，比如：默认, 必选

value 记录值,CDN返回的cname如CNAME: cname.dnspod.com,7nt6mrh7sdkslj.wscloudcdn.com, 必选

响应代码：

共通返回

-15 域名已被封禁

-7 企业账号的域名需要升级才能设置

-8 代理名下用户的域名需要升级才能设置

6 缺少参数或者参数错误

7 不是域名所有者或者没有权限

21 域名被锁定

22 子域名不合法

23 子域名级数超出限制

24 泛解析子域名错误

25 轮循记录数量超出限制

26 记录线路错误

27 记录类型错误

30 MX 值错误，1-20

31 存在冲突的记录(A记录、CNAME记录、URL记录不能共存)

32 记录的TTL值超出了限制

33 AAAA 记录数超出限制

34 记录值非法

36 @主机的NS纪录只能添加默认线路

82 不能添加黑名单中的IP

响应字段：

* id: 记录ID, 即为 record\_id
* name: 添加的子域名（即sub\_domain）
* status: 域名记录的状态

#### 设置记录状态

接口地址：

* https://dnsapi.cn/Record.Status

HTTP请求方式：

* POST

请求参数：

* 公共参数
* domain\_id 域名ID，必选
* record\_id 记录ID，必选
* status {enable|disable} 新的状态，必选

响应代码：

* 共通返回
* -15 域名已被封禁
* -7 企业账号的域名需要升级才能设置
* -8 代理名下用户的域名需要升级才能设置
* 6 域名ID错误
* 7 不是域名所有者或没有权限
* 8 记录ID错误
* 21 域名被锁定

响应字段：

* id: 记录ID, 即为 record\_id
* name: 子域名
* status: 记录状态

#### 获取记录信息

接口地址：

* https://dnsapi.cn/Record.Info

HTTP请求方式：

* POST

请求参数：

* 公共参数
* domain\_id 域名ID，必选
* record\_id 记录ID，必选

响应代码：

* 共通返回
* -15 域名已被封禁
* -7 企业账号的域名需要升级才能设置
* -8 代理名下用户的域名需要升级才能设置
* 6 域名ID错误
* 7 不是域名所有者或没有权限
* 8 记录ID错误

响应字段：

{

"status": {

"code": "1",

"message": "Action completed successful",

"created\_at": "2015-01-18 17:36:10"

},

"domain": {

"id": 2317346,

"domain": "testapi.com",

"domain\_grade": "D\_Plus"

},

"record": {

"id": "16909160",

"sub\_domain": "@",

"record\_type": "A",

"record\_line": "默认",

"value": "111.111.111.111",

"mx": "0",

"ttl": "10",

"enabled": "1",

"monitor\_status": "",

"remark": "test",

"updated\_on": "2015-01-18 17:23:58",

"domain\_id": "2317346"

}

}

字段说明:

domain:

id: 域名ID，即为 domain\_id

domain: 域名

domain\_grade: 域名等级，详见 Domain.List 或 Domain.Info 接口

record:

id: 记录ID编号

sub\_domain: 子域名(主机记录)

record\_type: 记录类型, 详见 Record.Type 接口

record\_line: 解析线路, 详见 Record.Line 接口

value: 记录值

mx: 记录的 MX 记录值, 非 MX 记录类型，默认为 0

ttl: 记录的 TTL 值

enabled: 记录状态

“0”: 禁用

“1”: 启用

monitor\_status: 该记录的D监控状态

“Ok”: 服务器正常

“Warn”: 该记录有报警, 服务器返回 4XX

“Down”: 服务器宕机

“”: 该记录未开启D监控

remark: 记录备注

updated\_on: 记录最后更新时间

domain\_id: 域名ID, 即为 domain\_id

#### 删除记录

接口地址：

* https://dnsapi.cn/Record.Remove

HTTP请求方式：

* POST

请求参数：

* 公共参数
* domain\_id 域名ID，必选
* record\_id 记录ID，必选

响应代码：

* 共通返回
* -15 域名已被封禁
* -7 企业账号的域名需要升级才能设置
* -8 代理名下用户的域名需要升级才能设置
* 6 域名ID错误
* 7 不是域名所有者或没有权限
* 8 记录ID错误
* 21 域名被锁定

### 本次新增的消息队列和计划任务

#### 消息队列

①、底层操作object信息队列**CDN\_REFRESH:SYNCOBJECT**

②、底层操作bucket信息队列**CDN\_REFRESH:SYNCBUCKET**

#### 计划任务

①、调用CDN清除缓存文件接口计划任务，每分钟一次，取队列信息执行。

②、执行底层删除bucket处理，每分钟一次，取队列信息执行。

③、调用CDN查询加速域名流量计划任务，每小时一次，取符合条件的bucket

④、每天计划任务

⑤、每周计划任务

⑥、每月计划任务

## Model设计

根据实际需求需要增加bucket和CDN的关联表，主要记录CDN的相关信息。同时将CDN和DNS调用的相关日志记入mongo中。

### CdnBucket

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CdnBucket | | | | |
| 功能说明 | 继承BaseCdnBucket | | | |
| 字段类型 | 字段名称 | 字段含义 | 默认 | 备注 |
| String | id | 记录id |  |  |
| String | cdnProvider | CDN提供商 |  | **enum**<UpYun, ChinaNetCenter> |
| String | cusId | 客户id |  |  |
| String | bucketName | Bucket名称 |  |  |
| Date | closeTime | 最后关闭时间 |  |  |
| Date | deleteTime | 删除时间 |  |  |
| String | isOpencdn | 是否开启CDN | 0 | 未开通：0  已开通：1 |
| String | cdnStatus | CDN状态 | 0 | 未加速：0  设置中：1  已加速：2 |
| String | cdnPath | CDN加速地址 |  | 网宿：CDN接口返回cname字段  UpYun：{bucket\_name }.file.eayun.com |
| String | domainId | 沟通API接口的唯一属性，加速域名标识 |  | 网宿API：CDN接口返回的唯一id；  UpYun：自己定义的用于沟通API的唯一bucket\_name |
| String | recordId | DNS记录id |  |  |
| String | isDelete | 是否删除 | 0 | 0：未删除  1：已删除 |

### BaseCdnBucket

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数据表名称** | | **cdn\_bucket** | | | |
| **功能说明** | | **记录bucket和CDN关联信息** | | | |
| **序号** | **字段名称** | **中文名称** | **类型** | **说明** | **业务规则** |
| 1 | id | Id | VC(36) |  |  |
| 2 | cdn\_provider | CDN提供商 | VC(36) |  |  |
| 3 | cus\_id | 客户id | VC(36) |  |  |
| 4 | bucket\_name | Bucket名称 | VC(64) |  |  |
| 5 | close\_time | 上次关闭时间 | Date |  |  |
| 6 | delete\_time | 删除时间 | Date |  |  |
| 7 | Is\_opencdn | 是否开通CDN | VC(10) |  |  |
| 8 | cdn\_status | Cdn状态 | VC(10) |  |  |
| 9 | cdn\_path | Cdn地址 | VC(255) |  |  |
| 10 | domain\_id | 加速域名标识 | VC(36) |  |  |
| 11 | record\_id | DNS记录id | VC(255) |  |  |
| 12 | is\_delete | 是否删除 | VC(10) |  |  |

### log.api.cdn

记录调用CDN接口的日志，存入mongo中，包括不同的提供商。

集合名为log.api.cdn，见mongo设计文档

log.api.cdn:{

"bucketName ":"" //bucket名称

"cdnProvider":"" //**enum**<UpYun,ChinaNetCenter>

"domainId":"" //域名标识，UpYun为自己生成的bucket\_name

"timestamp ":"" //操作时间

"operationCdn ":"" //CDN接口操作

" URL":"" //接口URL

"RequestBody":{}

"statuscode":""

"message":"" //错误信息，成功返回时不需要

"status ":"" //执行状态（0：失败；1：成功）

}

### log.api.dns

记录调用DNS接口的日志，存入mongo中

集合名为log.api.dns，见mongo设计文档

log.api.dns:{

"bucketName ":"" //bucket名称

"recordId":"" //DNS记录id

"domainId":"" //域名标识

"timestamp ":"" //操作时间

"operationDns ":"" //DNS接口操作

" URL":"" //接口URL（包含了业务参数信息）

"ResponseBody":{}

"status ":"" //执行状态（0：失败；1：成功），从ResponseBody中根据code判断

}

### ObsUsedType

增加两个字段**long** cdnFlow，**String** cdnFlowStr,用于显示CDN下载流量（资源统计）

### BucketStorageBean

增加一个字段BigDecimal cdnFlow，用于显示CDN下载流量（图表）

### ObsBucket

增加字段String cdnStatus（CDN状态），String isOpencdn（是否开启CDN），

String domainId（网宿为唯一的域名id，UpYun为生成的bucket\_name）

## Controller

由于本次增加CDN服务所需增加或修改的接口

### Object和Bucket操作变更通知接口

#### 包命、类名

提供该接口的Controller放在eayun-obs工程中，包命和类名如下：

com.eayun.obs.api.controller.ObsInternalAPIController

#### 请求地址和伪代码

请求地址格式如下：

http://<ip>:<port>/api/v1/obs/notice

伪代码如下：

@Controller  
@RequestMapping(**"/api"**)  
**public class**ObsInternalAPIController {

    @RequestMapping(method = RequestMethod.***POST***, value = **"/v1/obs/notice"**)

    @ResponseBody

**public**String objectNoticeV1(HttpServletRequest request, HttpServletResponse response, @RequestBody String requestBody) **throws**Exception {

*//1.*鉴权

     JSONObject result = **new**JSONObject();

        String date = request.getHeader(**"Date"**);

        String authorization = request.getHeader(**"Authorization"**);

        String customerId = request.getHeader(**"Customer"**);

**if**(date == **null**|| authorization == **null**|| customerId == **null**) {

            response.setStatus(401);

            result.put(**"keyword"**, **"Unauthorized"**);

            result.put(**"message"**, **"Access is denied due to invalid credentials"**);

**return**result.toJSONString();

        }

**boolean**isCustomerExist = isCustomerExist(customerId);

**if**(!isCustomerExist){

            response.setStatus(401);

            result.put(**"keyword"**, **"Unauthorized"**);

            result.put(**"message"**, **"No such customer id"**);

**return**result.toJSONString();

        }

        Date now = **new**Date();

        SimpleDateFormat sdf = **new**SimpleDateFormat(**"EEE, dd MMM yyyy HH:mm:ss z"**, Locale.***US***);

        sdf.setTimeZone(TimeZone.*getTimeZone*(**"GMT"**));

        Date headerDate = sdf.parse(date);

**long**timeOffset = 30\*60\*1000;

**long**dateCeiling = now.getTime() + timeOffset;

**long**dateFloor = now.getTime() - timeOffset;

**if**(headerDate.getTime() < dateFloor || headerDate.getTime() > dateCeiling){

            response.setStatus(400);

            result.put(**"keyword"**, **"BadRequest"**);

            result.put(**"message"**, **"Invalid date period"**);

**return**result.toJSONString();

        }

        String auth = authorization.substring(authorization.indexOf(**" "**)+1);

        String decodedAuth = **new**String(Base64.*decodeBase64*(auth.getBytes()), StandardCharsets.***UTF\_8***);

        String ak = decodedAuth.substring(0, decodedAuth.indexOf(**":"**)-1);

        String signature = decodedAuth.substring(decodedAuth.indexOf(**":"**)+1, decodedAuth.length());

        String sk = getSKByCustomerIdAndAK(ak, customerId);

        String recodedSignature = HmacSHA1Util.*getEncrypt*(date, sk);

**if**(!signature.equals(recodedSignature)){

            response.setStatus(401);

            result.put(**"keyword"**, **"Unauthorized"**);

            result.put(**"message"**, **"No such customer id"**);

**return**result.toJSONString();

        }

*//2.*组织参数放入任务队列

     JSONObject requestJSON = JSONObject.*parseObject*(requestBody);

        requestJSON.put(**"customerId"**, customerId);

        pushIntoTaskQueue(requestJSON);

        response.setStatus(200);

        result.put(**"keyword"**,**"Success"**);

        result.put(**"message"**,**"Task submitted successfully"**);

**return**result.toJSONString();

    }

@RequestMapping(method = RequestMethod.***DELETE***, value = **"/v1/obs/notice/{bucketName}"**)

@ResponseBody

**public**String bucketNotice(HttpServletRequest request, HttpServletResponse response, @PathVariable String bucketName) **throws**Exception {

    JSONObject result = **new**JSONObject();

*//1.*鉴权

*//2.*组织参数放入任务队列

  response.setStatus(200);

    result.put(**"keyword"**,**"Success"**);

    result.put(**"message"**,**"Task submitted successfully"**);

**return**result.toJSONString();

}

}

### ObsBucketController

#### getBucketPageList(getBucketPageList)OK

修改，在原方法的基础上通过查询关联表得到CDN状态

|  |  |
| --- | --- |
| 说明 | 查询客户未删除的bucket信息列表 |
| URL | /obs/bucket/getBucketPageList.do |
| 方式 | post |
| Post数据 | {  Page:{}  "name":""//bucket名称  } |
| 返回Json | {  Page:{}  "data":[  {  “bucketName”:"", //bucket名称  “creationDate”:"", //创建时间  “permission”:"" //bucket权限  "permissionEn":"" //bucket权限  "cdnStatus":"" //CDN加速域名状态  "isOpencdn":"" //是否开启CDN  "domainId":"" //CDN的域名标识  详细参见ObsBucket对象  },… …  ]  } |
|  |  |

### ObsCdnBucketController（新增）

#### openCDN(opencdn)

开启CDN服务,新增接口

|  |  |
| --- | --- |
| 说明 | 开启CDN |
| 备注 |  |
| URL | /obs/cdn/opencdn.do |
| 方式 | post |
| Post数据 | {  " bucketName":""//选中的bucket的名称  } |
| 返回Json | {  "respCode":"000000",  }  ----失败--------------------------------------------------  {  "respCode":"010120",  } |

#### closeCDN(closecdn)

关闭CDN服务，新增接口

|  |  |
| --- | --- |
| 说明 | 关闭CDN |
| 备注 |  |
| URL | /obs/cdn/ closecdn.do |
| 方式 | post |
| Post数据 | {  " bucketName":""//选中的bucket的名称  } |
| 返回Json | {  "respCode":"000000",  }  ----失败--------------------------------------------------  {  "respCode":"010120",  } |

#### getCdnFlowByBucket(getcdnflowbybucket)

获取bucket的CDN下载流量（详情页图表），新增接口

|  |  |
| --- | --- |
| 说明 | 获取详情页图表的CDN下载流量 |
| 备注 | 首先查询关联表中是否存在未删除的记录 |
| URL | /obs/cdn/getcdnflowbybucket.do |
| 方式 | post |
| Post数据 | {  "bucketName":""//选中的bucket的名称  } |
| 返回Json | {  "respCode":"000000",  "data":[  {  " cdnFlow ":""//下载流量  "timestamp":""//采集时间  },  {  " cdnFlow ":""  "timestamp":""  }......  ]  }  ----失败--------------------------------------------------  {  "respCode":"010120",  } |

### ObsUsedController

#### getObsResources(getObsResources)OK

对象存储资源统计接口，修改原来的方法，增加CDN下载流量统计

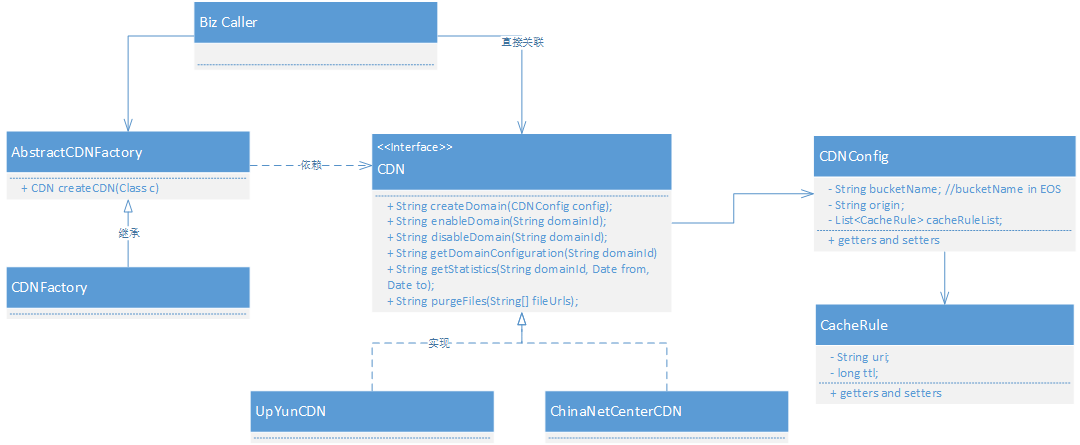
|  |  |
| --- | --- |
| 说明 | 客户对象存储资源统计报表 |
| 备注 | 增加了CDN下载流量 |
| URL | /obs/used/getObsResources.do |
| 方式 | post |
| Post数据 | {  "startTime":""  "endTime":""  } |
| 返回Json | {  "respCode":"000000",  "data":[  {  " requestCount":""//请求次数  "loadDown":""//下载流量  "usedStorage":""//存储容量  " cdnFlowStr ":""//CDN下载流量  .......  },  ......详见ObsUsedType对象  ]  }  ----失败--------------------------------------------------  {  "respCode":"010120",  } |

## Service接口

由于增加CDN服务所需增加或修改的接口

### CDN接口封装

一、类图



上面的类图中，抽象类AbstractCDNFactory定义了CDN的整体功能，CDNFactory为实现类，具体实现了创建CDN的任务；接口CDN定义了CDN的整体功能，分别有UpYunCDN和ChinaNetCenterCDN两个实现类，其中，还关联了CDNConfig这个bean，CDNConfig中包含一个CacheRule的列表。

二、代码

对于AbstractCDNFactory，我们采用了泛型，因为对于工厂类，只需要传入CDN的实现类就可以了，使用泛型，既约束了参数类型，又减少了对象间的转换。

**public abstract class**AbstractCDNFactory {  
    **public abstract**<T **extends**CDN> T createCDN(Class<T> c);  
}

对于CDNFactory，我们采用了反射的机制，这样我们不需要在工厂类中创建对象实例，而是交给对象自己去创建（Object java.lang.Class.newInstance()）：

**public class**CDNFactory **extends**AbstractCDNFactory{

  @Override

**public**<T **extends**CDN> T createCDN(Class<T> c) {  
        CDN cdn = **null**;  
        **try**{  
            cdn = (T)Class.*forName*(c.getName()).newInstance();  
        } **catch**(Exception e) {  
            e.printStackTrace();  
        }  
        **return**(T)cdn;  
    }  
}

在业务调用者这里，我们为其封装一个工具类：

**public class** CDNUtil {  
**public static** CDN getUpYunCDN(){  
 CDNFactory cdnFactory = **new** CDNFactory();  
 CDN upYunCDN = cdnFactory.createCDN(UpYunCDN.**class**);  
**return** upYunCDN;  
 }  
}

由业务调用者直接调用CDNUtil.getUpYunCDN()即可获得UpYunCDN的实例，然后进行后续操作。

三、定义

看完上面的类图和代码，是时候引入工厂方法模式的定义了：

Factory method pattern--"Define an interface for creating an object, but let subclasses decide which class to instantiate. The Factory method lets a class defer instantiation it uses to subclasses"

即，定义一个用于创建对象的接口，让子类决定实例化哪个类。工厂方法使一个类的实例化延迟到其子类。

四、应用及扩展

工厂方法模式具备良好的封装性，代码结构清晰。当调用者需要一个具体的产品对象，只需要知道产品的类名（或约束字符串）即可，不需要了解具体创建过程，降低了模块件的耦合性。

在增加产品类的情况下，只需要修改工厂类或者扩展一个工厂类即可，在本例中，我们不需要修改任何工厂类即可完成扩展。所以扩展性十分优秀。

此外，工厂方法还屏蔽掉了产品类，无论产品类的实现如何变化，调用者都不需要关心，只要接口保持不了，调用者的上层模块就不需要发生变化。

当我们去掉抽象类AbstractCDNFactory，同时把CDNFactory中的createCDN方法设置为静态方法，则简化后的代码就形成了简单工厂方法。

当我们增加多个工厂类，如UpYunCDNFactory、ChinaNetCenterCDNFactory，由调用者选择使用哪个工厂类去创建与其关联的产品，此时非常符合单一职责原则，这就是多工厂类的工厂方法模式。在这种模式扩展一个产品类的时候，需要对应创建一个工厂类，略显复杂。不过在复杂应用中，都会采用多工厂模式，然后增加一个协调类，对上层调用者屏蔽个各个工厂类，从而提供一个统一的访问接口。个人认为我们这里还不至于复杂到需要多工厂的情况。

### ObsBucketService

#### getBucketPage

//原方法查询出bucket列表后，关联查询关系表获得CDN相关状态。

### ObsUsedService

#### getObsUsedList

//增加查询CDN下载流量的数据

### obsBaseStorageService

#### getUrl

//修改原来的方法，开启CDN时获取加速地址，未开启时获取原地址

CdnBucket cdnBucket = obsCdnBucketService.getOpenByName(bucketName);

**if**(**null** != cdnBucket){

String post = getCdnUrlByNodeID("0009005");

client.setEndpoint(post);

}

### ObsCdnBucketService（新增）

#### enableDomain

/\*\*

\* 开启CDN服务

\* **@Author**: duanbinbin

\* **@param** bucketName

\* **@param** cusId

\* **@param** cdnProvider

\*<li>Date: 2016年6月16日</li>

\*/

**publicboolean** enableDomain(String bucketName , String cusId, String cdnProvider);

#### disableDomain

/\*\*

\* 关闭CDN服务

\* **@Author**: duanbinbin

\* **@param** bucketName

\* **@param** cusId

\* **@param** cdnProvider

\*<li>Date: 2016年6月16日</li>

\*/

**publicboolean** disableDomain(String bucketName , String cusId, String cdnProvider);

#### getCDNFlowData

/\*\*

\* 查询bucket的CDN下载流量

\* **@Author**: duanbinbin

\* **@param** bucketName

\* **@return**

\*<li>Date: 2016年6月16日</li>

\*/

**public** List<BucketStorageBean> getCDNFlowData(String bucketName);

#### getUnDeleteListByCusId

/\*\*

\* 查询客户所有未删除的关联记录

\* **@Author**: duanbinbin

\* **@param** cusId

\* **@return**

\*<li>Date: 2016年6月16日</li>

\*/

**public** List<CdnBucket> getUnDeleteListByCusId(String cusId);

#### getUnDeleteByCusAndName

/\*\*

\* 查询指定客户下指定bucket是否有未删除的关联记录

\* **@Author**: duanbinbin

\* **@param** bucketName

\* **@param** cusId

\* **@return**

\*<li>Date: 2016年6月16日</li>

\*/

**public** CdnBucket getUnDeleteByCusAndName(String bucketName , String cusId , String cdnProvider);

#### addCdnLog（操作CDN时调用）

/\*\*

\* 添加CDN日志

\* **@param** bucketName bucket名称

\* **@param** doamin 域名标识

\* **@param** url 接口URL

\* **@param** operationCdn CDN接口操作

\* **@param** requestBody 接口请求体，包含业务参数信息

\* **@param** statuscode 请求状态

\* **@param** result 返回json，包含result（成功true，失败false）和message（失败详情）

\*<li>Date: 2016年6月16日</li>

\*/

**publicvoid** addCdnLog(String bucketName,String doamin,String url,String operationCdn ,

JSONObject requestBody ,**int** statuscode,JSONObject result);

#### addDnsLog（操作DNS时调用）

/\*\*

\* 添加DNS日志

\* **@Author**: duanbinbin

\* **@param** bucketName bucket名称

\* **@param** domain 域名标识

\* **@param** url DNS接口 URL（包含了业务参数）

\* **@param** operationDns DNS接口操作

\* **@param** responseBody 所有原生返回信息（里面code为1时，表示请求成功）

\* **@param** recordId DNS记录

\*<li>Date: 2016年6月16日</li>

\*/

**publicvoid** addDnsLog(String bucketName,String domain,String url,

String operationDns ,JSONObject responseBody ,String recordId);

## 增加计划任务

### CdnDetailGatherThread

//1.每小时02分执行，查询上一个整小时内的加速域名的下载流量

//2.查询关联表里所有开通过CDN的bucket记录列表；条件：①已开启或（未开启&&上次关闭时间不超过两个小时）。

//3.根据查询出的bucket的domain调用CDN的查询流量接口，条件：domain-id，datefrom ，dateto，type = fiveminutes

//4.按bucket记录流量数据

//5.记录此次查询操作的CDN日志

### CdnDaySummaryThread

### CdnWeekSummaryThread

### CdnMonthSummaryThread

### CdnRefreshObjThread

### CdnRefreshBucketThread