金山云 CDN
API 使用手册
刷新预加载接口
V4.0

北京金山云网络技术有限公司

2016-05-10

目录

1 简介

金山云 CDN API 是一套提供给客户方便运用代码更加快捷使用 CDN 的 API 文档。客户通过调取 API 可对金山云 CDN 进行函数调用，方便让客户的开发人员在不需要理解系统架构或访问源代码的同时获得访问例程的能力。

2 调用方式

金山云 CDN 的 API 接口通过 HTTP 方式对其进行调用，在调用 API 之前，需要使用到 AccessKey 和 SecretKey 进行鉴权。其中，AccessKey 用来进行身份识别，加上 SecretKey 进行数字签名，即可完成应用接入与认证授权。

3 鉴权方式

3.1 创建规范请求

在创建签名之前，需要创建规范的请求，将请求设置为规范的格式，规范请求伪代码：

```
CanonicalRequest =
  HTTPRequestMethod + '\n' +
  CanonicalURI + '\n' +
  CanonicalQueryString + '\n' +
  CanonicalHeaders + '\n' +
  SignedHeaders + '\n' +
  HexEncode(Hash(RequestPayload))
```

在此伪代码中，Hash 表示生成消息摘要的函数，通常是 SHA-256。HexEncode 表示以小写字母形式返回摘要的 base-16 编码的函数。例如，HexEncode("m") 返回值 6d 而不是 6D。输入的每一个字节都必须表示为两个十六进制字符。

示例请求

```
GET https://cdnapi.ksyun.com/?Action=ListUsers&Version=2010-05-08
HTTP/1.1
Host:cdnapi.ksyun.com
Content-Type: application/xml; charset=utf-8
X-Amz-Date: 20150830T123600Z
```

创建规范请求，需要将以下步骤连接成一个字符串：

1）http 请求方法(POST、GET、PUT)，后跟换行符。

示例：请求方法

```
POST
```

2）添加规范 URL 参数，后跟换行符。规范 URI 是 URI 的绝对路径部分的 URI 编码版本，该版本是 URI 中的一切 - 从 HTTP 主机到开始查询字符串参数（如果有）的问号字符（“?”）。

根据 RFC 3986，通过移除冗余和相对路径部分标准化 URI 路径。路径中每个部分都必须为 URI 编码。

使用编码的示例规范 URI

```
%2Fhome%2Fdocuments%20and%20settings
```

如果绝对路径为空，则使用正斜杠 (/)。在示例请求中，URI 中的主机后没有任何内容，因此，绝对路径为空。

示例：规范 URI

```
/
```

3）添加规范查询字符串，后跟换行符。如果请求不包含查询字符串，请在规范查询中将此值设置为空字符串(实际上是空白行)。示例请求具有以下查询字符串。

示例：规范查询字符串

```
Action=ListUsers&Version=2010-05-08
```

要构建规范查询字符串，请完成以下步骤：

a)根据以下规则对每个参数名称和值进行 URI 编码：

请勿对 RFC 3986 定义的任何非预留字符进行 URI 编码，这些字符包括：A-Z、a-z、0-9、连字符 (-)、下划线 (_)、句点 (.)) 和波浪符 (~)。

使用 %XY 对所有其他字符进行百分比编码，其中“X”和“Y”为十六进制字符（0-9 和大写字母 A-F)。

例如，空格字符必须编码为 %20（不像某些编码方案那样使用“+”），扩展 UTF-8 字符必须采用格式 %XY%ZA%BC。

b)按字符代码以升序（ASCII 顺序）对编码后的参数名称进行排序。例如，以大写字母 F（ASCII 代码 70）开头的参数名称排在以小写字母 b（ASCII 代 码 98）开头的参数名称之前。

c)以排序后的列表中第一个参数名称开头，构造规范查询字符串。

d)对于每个参数，追加 URI 编码的参数名称，后跟字符“=”（ASCII 代码 61)， 再接 URI 编码的参数值。对没有值的参数使用空字符串。

e)在每个参数值后追加字符“&”（ASCII 代码 38)，列表中最后一个值除外。

查询 API 的一种方案是将所有请求参数放入查询字符串中，在这种情况下，规范查询字符串不能只包含请求的参数,还必须包含在签名流程中要使用的参数 - 哈希算法、凭证范围、日期和已签名标头参数。

下面的示例说明包含身份验证信息的查询字符串

```
Action=ListUsers&
Version=2010-05-08&
X-Amz-Algorithm=AWS4-HMAC-SHA256&
X-Amz-Credential=AKIDEXAMPLE%2F20150830%2Fus-east-
1%2Fcloudfront%2Faws4_request&
X-Amz-Date=20150830T123600Z&
X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
```

以上代码为了便于阅读，添加了换行符，规范查询字符串必须是连续的一行。

4）添加规范标头，后跟换行符。规范标头包括您要包含在请求中的所有 HTTP 标头的列表。

示例：规范标头

```
content-type:application/xml; charset=utf-8\n
host:cdnapi.ksyun.com\n
```

```
x-amz-date:20150830T123600Z\n
```

要创建规范标头列表，请将所有标头名称转换为小写，并从标头值中去除多余的空白字符。去除这些字符时，请删除值前后的空格，将值中间的连续空格转变为单个空格。但是，不要删除引号内任何值中的多余空格。

以下伪代码描述如何构造规范标头列表：

```
CanonicalHeaders =
CanonicalHeadersEntry0 + CanonicalHeadersEntry1 + ... +
CanonicalHeadersEntryN
CanonicalHeadersEntry =
Lowercase(HeaderName) + ':' + Trimall(HeaderValue) + '\n'
```

Lowercase 表示将所有字符转换为小写字母的函数。Trimall 函数删除值前后和不在引号内的字符串中的多余空格。

通过按小写字符代码对标头进行排序，然后对标头名称集合进行迭代操作，来构建规范标头列表。根据以下规则构造每个标头：

追加小写标头名称，后跟冒号。

追加该标头的值的逗号分隔列表。如果存在重复标头，则用逗号分隔标头值。请勿对有多个值的标头进行值排序。

追加一个新行 (“\n”)。

注意：

下列示例对更复杂的一组标头及其规范形式进行比较：

原始标头：

```
Host:cdnapi.ksyun.com\n
Content-Type:application/x-www-form-urlencoded; charset=utf-8\n
My-header1:  a  b  c \n
X-Amz-Date:20150830T123600Z\n
My-Header2:  "a  b  c"\n
```

规范标头：

```
content-type:application/x-www-form-urlencoded; charset=utf-8\n
host:cdnapi.ksyun.com\n
my-header1:a b c\n
my-header2:"a  b  c"\n
x-amz-date:20150830T123600Z\n
```

注意：

每个标头都后跟换行符，这意味着完整列表以换行符结束。

对于规范形式，进行了下列更改：

标头名称已转换为小写字符。

标头已按字符代码进行排序。

已从 my-header1 的值中删除多余空格，因为该值不在引号内。

已删除 my-header1 和 my-header2 值前面的空格。未从 my-header2 的值中删除多余空格，因为该值在引号内。

5）添加已签名的标头，后跟换行符。该值是您包含在规范标头中的标头名称的列表。

host 标头必须作为已签名标头包括在内。如果包括日期或 x-amz-date 标头，

则还必须包括在已签名标头列表中的标头。

要创建已签名标头列表，请将所有标头名称转换为小写形式，按字符代码对其进行排序，并使用分号来分隔这些标头名称。以下伪代码描述如何构建已签名标头的列表。LOWERCASE 表示将所有字符转换为小写字母的函数。

```
SignedHeaders =
Lowercase(HeaderName0) + ';' + Lowercase(HeaderName1) + ";" + ... +
Lowercase(HeaderNameN)
```

通过对按小写字符代码排序的标头名称集合进行迭代操作，构建已签名标头的列表。对于除最后一个标头外的每个标头名称，请在标头名称后追加分号(“;”)，将它与后面的标头名称分隔开。

示例：已签名的标头

```
content-type;host;x-amz-date\n
```

6）使用 SHA256 等哈希（摘要）函数以基于 HTTP 或 HTTPS 请求正文中的负载创建哈希值。

负载结构

```
HashedPayload = Lowercase(HexEncode(Hash(requestPayload)))
```

在创建待签字符串后，请指定用于对负载进行哈希处理的签名算法。例如，如果您使用的是 SHA256，则将指定 AWS4-HMAC-SHA256 作为签名算法。经过哈希处理的负载必须以小写十六进制字符串形式表示。

如果负载为空，则使用空字符串作为哈希函数的输入。在此 cloudfront 示例中，负载为空。

示例：经过哈希处理的负载(空字符串)

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

7）要构建完整的规范请求，请将来自每个步骤的所有组成部分组合为单个字符串。如上所述，每个组成部分都以换行符结尾。如果您执行前述规范请求伪代码，生成的规范请求将显示在以下示例中。

示例：规范形式

```
GET
/
Action=ListUsers&Version=2010-05-08
content-type:application/x-www-form-urlencoded; charset=utf-8
host:cdnapi.ksyun.com
x-amz-date:20150830T123600Z
content-type;host;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

8）使用您对负载进行哈希处理时所使用的相同算法来创建规范请求的摘要（哈希）。

经过哈希处理的规范请求必须以小写十六进制字符字符串形式表示。以下示例显示了使用 SHA-256 对示例规范请求进行哈希处理的结果。

示例：经过哈希处理的规范请求

```
f536975d06c0309214f805bb90ccff089219ecd68b2577efef23edd43b7e1a59
```

3.2 创建待签字字符串

待签字符串包含请求及规范请求的元信息。将待签字符串及计算签名作为签名密钥。

要创建待签字符串，请如以下伪代码所示，连接算法、日期、证书范围和规范请求的摘要：

待签字符串结构：

```
StringToSign  =
Algorithm + '\n' +
RequestDate + '\n' +
CredentialScope + '\n' +
HashedCanonicalRequest))
```

以下示例演示如何使用规范请求中的相同请求构造待签字符串。

示例 HTTP 请求：

```
GEThttps://cdnapi.ksyun.com/?Action=ListUsers&Version=2010-05-08
HTTP/1.1
Host: cdnapi.ksyun.com
Content-Type: application/xml; charset=utf-8
X-Amz-Date: 20150830T123600Z
```

创建待签字符串

1）以算法名称开头，后跟换行符。该值是您用于计算规范请求摘要的哈希算法。对于 SHA256，算法是 AWS4-HMAC-SHA256。

```
AWS4-HMAC-SHA256\n
```

2）追加请求日期值，后跟换行符。该日期是使用 ISO8601 基本格式以 YYYYMMDD'T'HHMMSS'Z' 格式在 x-amz-date 标头中指定的。此值必须与您在 前面所有步骤中使用的值匹配。

```
20150830T123600Z\n
```

3）追加证书范围值，后跟换行符。此值是一个字符串，包含日期、目标区域、所请求的服务和小写字符形式的终止字符串（“aws4_request”）。区域和服务名称字符串必须采用 UTF-8 编码。

```
20150830/us-east-1/cloudfront/aws4_request\n
```

日期必须为 YYYYMMDD 格式。请注意，日期不包括时间值。

有关区域列表，请参阅区域和终端节点。请验证您在此处指定的区域是您发送请求的目标区域。

4）追加规范请求的哈希规范其你去中创建的规范请求的哈希。该值后边不跟换行符，经过哈希处理的规范请求必须为 base-16 编码的小写形式。

```
f536975d06c0309214f805bb90ccff089219ecd68b2577efef23edd43b7e1a59
```

以下待签字符串是 2015 年 8 月 30 日的对 CDN API 的请求。

示例待签字符串

```
AWS4-HMAC-SHA256
20150830T123600Z
20150830/us-east-1/cloudfront/aws4_request
f536975d06c0309214f805bb90ccff089219ecd68b2577efef23edd43b7e1a59
```

3.3 计算签名

在计算签名之前，从私有访问密钥派生出签名密钥。由于生成签名密钥特定于日期、服务和区域，因此它提供了更高程度的保护。秘密访问密钥不只用于对请求进行签名。然后将签名密钥和带签字符串用作加密哈希函数的输入。加密哈希函数生成的十六进制编码结果就是签名。

计算签名

1）生成您的签名密钥。为此，请使用您的秘密访问密钥创建一系列基于哈希的消息身份验证代码 (HMAC)。此代码显示在以下伪代码中，其中 HMAC(key, data) 表示以二进制格式返回输出的 HMAC-SHA256 函数。每个哈希函数的结果将成为下一个函数的输入。

用于生成签名密钥的伪代码

```
kSecret = Your Access Key
kDate = HMAC("AWS4" + kSecret, Date)
kRegion = HMAC(kDate, Region)
kService = HMAC(kRegion, Service)
kSigning = HMAC(kService, "aws4_request")
```

注意，哈希过程中所使用的日期的格式为 YYYYMMDD（例如，20150830），不包括时间。

确保以正确的顺序为您要使用的编程语言指定 HMAC 参数。在此示例中，密钥是第一个参数，内容是第二个参数，但您使用的函数可能以不同顺序指定密钥和内容。

使用摘要来生成密钥。大多数语言都有用来计算二进制格式哈希（通常称为摘要）或十六进制编码哈希（称为十六进制摘要）的函数。生成密钥需要您使用摘要。

以下示例显示了用于生成签名密钥的输入以及所生成的输出，其中 kSecret =wJalrXUtnFEMI/K7MDENG+bPxRfiCYEXAMPLEKEY。

示例输入

```
HMAC(HMAC(HMAC(HMAC("AWS4" + kSecret,"20150830"),"us-east-
1"),"cloudfront"),"aws4_request")
```

以下示例显示了此 HMAC 哈希操作序列生成的派生签名密钥。这说明了此二进制派生签名密钥中每个字节的整数表示形式。

示例签名密钥

```
196 175 177 204 87 113 216 113 118 58 57 62 68 183 3 87 27 85 204 40 66
77 26 94 134 218 110 211 193 84 164 185
```

2）计算签名。要计算签名，请使用派生的签名密钥和待签字符串作为加密 哈希函数的输入。在计算摘要形式的签名后，将二进制值转换为十六进制表示 形式。以下伪代码说明如何计算签名。

```
signature = HexEncode(HMAC(derived-signing-key, string-to-sign))
```

以下示例显示了使用相同的签名密钥和待签字符串会生成的签名：

示例签名

```
5d672d79c15b13162d9279b0855cfba6789a8edb4c82c400e06b5924a6f2b5d7
```

3.4 将签名信息添加到请求

在计算签名后，将它添加到请求。您可以通过两种方式将签名信息添加到请求：

1）在名为 Authorization 的标头中。

2）在查询字符串中。

不能同时在 Authorization 标头和查询字符串中传递签名信息。

将签名信息添加到 Authorization 标头

通过将签名信息添加到名为 Authorization 的标头，可以包括签名信息。此标头内容是在按前面的步骤所述计算签名之后创建的，因此 Authorization 标头未包含在已签名标头的列表中。尽管此标头名为 Authorization，但签名信息实际上用于身份验证。

以下伪代码说明 Authorization 标头的构造。

```
Authorization: algorithm Credential=access key ID/credential scope,
SignedHeaders=SignedHeaders, Signature=signature
```

完成的 Authorization 标头示例

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIDEXAMPLE/20150830/us-
east-1/cloudfront/aws4_request, SignedHeaders=content-type;host;x-amz-
date,
Signature=5d672d79c15b13162d9279b0855cfba6789a8edb4c82c400e06b5924a6f2b
5d7
```

请注意以下几点：

1）算法与 Credential 之间有一个空格，而不是逗号。但是，SignedHeaders 和 Signature 值均用逗号与之前的值隔开。

2）Credential 值以访问密钥 ID 开头，后跟一个斜杠 (/)，再接带签字符串中计算得出的凭证范围值。秘密访问密钥用于为签名派生签名密钥，但未包含在通过请求发送的签名信息中。

将签名信息添加到查询字符串

您可以发送请求并在查询字符串中传递所有请求值，包括签名信息。这有时称为预签名 URL，因为它生成单个 URL，其中包含成功调用 API 所需要的一切信息。

备注

如果您发出一个请求，其中所有参数都包含在查询字符串中，则生成的 URL 表示已经进行了身份验证的操作。因此，对待生成的 URL 要像对待实际凭证一样小心。建议您使用 X-Amz-Expires 参数为请求指定较短的过期时间。

如果使用这种方法,所有查询字符串值(签名除外)都将包含在规范查询字符串中，该字符串是您在签名过程第一部分中构造的规范查询的一部分。

以下伪代码说明包含所有请求参数的查询字符串的构造。

```
querystring = Action=action
querystring += &X-Amz-Algorithm=algorithm
querystring += &X-Amz-Credential= urlencode(access_key_ID + '/' +
credential_scope)
```

```
querystring += &X-Amz-Date=date
querystring += &X-Amz-Expires=timeout interval
querystring += &X-Amz-SignedHeaders=signed_headers
```

在计算签名（使用其他查询字符串值作为计算的一部分）后，请将该签名作为 X-Amz-Signature 参数添加到查询字符串中：

```
querystring += &X-Amz-Signature=signature
```

下面的示例说明当所有请求参数（包括签名信息）都包括在查询字符串参数中时请求看起来是什么样。

```
https://cdnapi.ksyun.com?Action=ListUsers&Version=2010-05-08&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIDEXAMPLE%2F20150830%2Fus-east-1%2Fcloudfront%2Faws4_request&X-Amz-Date=20150830T123600Z&X-Amz-Expires=60&X-Amz-SignedHeaders=content-type%3Bhost&X-Amz-Signature=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d181d02
```

请注意以下几点：

为计算签名，查询字符串参数必须为 ASCII 顺序，并且其值必须为 URI 编码。有关详细信息，请参阅 创建规范请求中有关创建规范查询字符串的步骤。

将超时间隔 (X-Amz-Expires) 设置为您所请求的操作的最短可行时间。

每个接口请求示例中的 x-action 和 x-version 请严格按照文档给出的进行填写。

4 刷新 API

4.1 API 描述

刷新接口可分别对 URL 和目录进行刷新，URL 刷新上限为 1000 条/次，目录刷新上限为 100 条/次。

4.2 请求示例

4.2.1 语法

```
POST /2015-09-17/distribution/{distribution}/invalidation
```

4.2.2 请求参数

名称	描述	备注	必须
Quantity	刷新的数目		是
Path	刷新的 URL 或目录		是

4.2.3 请求示例

```
POST/2015-09-17/distribution/d3d3LmJhaWR1LmNvbQ==/invalidation
HTTP/1.0
Host: cdnapi.ksyun.com
Authorization: AWS authentication string
```

```
Content-Type: text/xml
x-action: CreateInvalidation
x-version: 2015-09-17
<InvalidationBatch >
   <Paths>
      <Quantity>3</Quantity>
      <Items>
         <Path>/image1.jpg</Path>
         <Path>/image2.jpg</Path>
         <Path>/videos/</Path>
      </Items>
   </Paths>
   <CallerReference>20120301090001</CallerReference>
</InvalidationBatch>
```

注释：

POST：d3d3LmJhaWR1LmNvbQ== 即目标请求的地址，生成方式为域名 base64 编码；

Host：API 请求的地址；

Content-Type：文件类型，不可改；

Path：刷新的目录名称，要求均以“/”为结尾；

CallerReference：用户请求标识，随机生成，要求每个不重复即可。

4.3 响应示例

```
HTTP/1.0 201 Created
Content-Type: text/xml
<Invalidation>
   <Id>IDFDVBD632BHDS5</Id>
   <Status>InProgress</Status>
   <CreateTime>2009-11-19T19:37:58Z</CreateTime>
   <InvalidationBatch>
      <Paths>
         <Quantity>3</Quantity>
         <Items>
            <Path>/image1.jpg</Path>
            <Path>/image2.jpg</Path>
            <Path>/videos/</Path>
         </Items>
      </Paths>
      <CallerReference>20120301090001</CallerReference>
   </InvalidationBatch>
</Invalidation>
```

注释：

ID：本次请求的 ID；

CreateTime：本次请求的创建时间；

5 预加载 API

5.1 API 描述

预加载接口可对文件进行预加载操作，上限为 100 条/次。

5.2 请求示例

5.2.1 语法

```
POST / 2015-09-17/distribution/{distribution}/preload
```

5.2.2 请求参数

名称	描述	备注	必须
Quantity	预加载的数目		是
Path	预加载的 URL		是

5.2.3 请求示例

```
POST /2015-09-17/distribution/d3d3LmJhaWR1LmNvbQ==/preload
HTTP/1.0
Host: cdnapi.ksyun.com
Authorization: authentication string
Content-Type: text/xml
x-action: CreatePreload
x-version: 2015-09-17
<?xml version="1.0" encoding="UTF-8"?>
<PreloadBatch>
<Paths>
<Quantity>3</Quantity>
<Items>
<Path>/info_01.html</Path>
<Path>/info_02.html</Path>
<Path>/info_03.html</Path>
</Items>
</Paths>
<CallerReference>10002</CallerReference>
</PreloadBatch>
```

5.3 响应示例

```
HTTP/1.0 201 Created
Content-Type: text/xml
<?xml version="1.0" encoding="UTF-8"?>
<Preload>
    <Id>9fe09cab-c073-434d-8252-471e86c15146</Id>
```

```
    <Status>InProgress</Status>
    <CreateTime>2016-01-18T04:08:51.913Z</CreateTime>
    <PreloadBatch>
        <Paths>
            <Quantity>3</Quantity>
            <Items>
                <Path>/info_01.html</Path>
                <Path>/info_02.html</Path>
                <Path>/info_03.html</Path>
            </Items>
        </Paths>
        <CallerReference>10002</CallerReference>
    </PreloadBatch>
</Preload>
```

注释：

Status:状态，公有两种状态：InProgress 处理中；Completed 完成；Failed 失败；

6 刷新预加载查询 API

6.1 API 描述

对刷新和预加载操作进行历史查询，可以查询规定时间内刷新的文件、目录、预加载的记录。

6.2 请求示例

6.2.1 语法

```
GET/2015-09-17/distribution/content-
path?QueryName=wangwei&Type=refreshDir&StartTime=1448973434854&EndTime=
1457613434857&PageIndex=1&PageSize=10
```

6.2.2 请求参数

名称	描述	备注	必须
QueryName	查询的内容		否
Type	查询的类型， refreshFile(刷新文件) refreshDir(刷新目录) preloadFile(预加载)		是
StartTime	开始时间	时间戳	否
EndTime	结束时间	时间戳	否
PageIndex	第几页	从 0 开始	否
PageSize	页大小	默认 100	否

6.2.3 请求示例

```
GET/2015-09-17/distribution/content-
path?QueryName=wangwei&Type=refreshDir&StartTime=1448973434854&EndTime=
1457613434857&PageIndex=1&PageSize=10
HTTP/1.0
Host: cdnapi.ksyun.com
Authorization: authentication string
Content-Type:application/xml
x-action: ListInvalidationsByContentPath
x-version: 2015-09-17
```

6.3 响应示例

```
<?xml version="1.0" encoding="UTF-8"?>
  <ContentPathList>
    <UserId> your userid </UserId>
    <QueryName>wangwei</QueryName>
    <Type>refreshDir</Type>
    <StartTimeMillis>1448973434854</StartTimeMillis>
    <EndTimeMillis>1457613434857</EndTimeMillis>
    <PageIndex>1</PageIndex>
    <PageSize>10</PageSize>
    <Quantity>3</Quantity>
    <Total>13</Total>
    <Items>
      <ContentPath>
        <Url>http://dl3.caohua.com/wangweitest/info_04/</Url>
        <Status>Failed</Status>
        <CreateTime>2016-02-24T13:12:58.000Z</CreateTime>
      <UpdateTime>2016-02-24T13:20:01.000Z</UpdateTime>
      </ContentPath>
    </Items>
  </ContentPathList>
```

注释：

QueryName：查询内容；

StartTimeMillis：开始时间戳；

Url：刷新/预加载请求的 URL；

7 查询刷新任务列表 API

7.1 API 描述

对该用户下指定域名的全部刷新任务查询。结果根据时间倒序排序。

7.2 请求示例

7.2.1 语法

```
GET
/2015-09-
17/distribution/{DistributionId}/invalidation?Marker={Marker}&MaxItems={MaxItem
s}
```

7.2.2 请求参数

名称	描述	备注	必须
marker	指定任务查询的开始位置	输入要查询开始的任务 ID，ID 按倒序排序；	否
MaxItems	每一页显示的最大任务个数	为空时默认 10	否
DistributionId	域名的 base64 编码		是

7.2.3 请求示例

```
GET
/2015-09-
17/distribution/{DistributionId}/invalidation?Marker={Marker}&MaxItems={MaxItem
s}
HTTP/1.0
Host: cdnapi.ksyun.com
Authorization: authentication string
Content-Type:application/xml
x-action: ListInvalidations
x-version: 2015-09-17
```

7.3 响应示例

```
<?xml version="1.0" encoding="UTF-8"?>
<InvalidationList>
   <MaxItems>10</MaxItems>
   <IsTruncated>false</IsTruncated>
   <Quantity>3</Quantity>
   <Items>
      <InvalidationSummary>
         <Id>bfe69949-3fc4-4c7a-8c03-4d4ceba86627</Id>
         <CreateTime>2016-04-06T09:59:15.000Z</CreateTime>
         <Status>Completed</Status>
      </InvalidationSummary>
      <InvalidationSummary>
         <Id>3e4b66f8-cbf4-4ba6-95b0-3df3da24be28</Id>
```

```
        <CreateTime>2016-04-05T13:42:37.000Z</CreateTime>
        <Status>InProgress</Status>
      </InvalidationSummary>
      <InvalidationSummary>
        <Id>3e4b66f8-cbf4-4ba6-95b0-3df3da24be29</Id>
        <CreateTime>2016-04-05T13:42:37.000Z</CreateTime>
        <Status>Failed</Status>
      </InvalidationSummary>
    </Items>
</InvalidationList>
```

注释：

IsTruncated： 判断是否有下一页；

NextMarker： 下一页开始的位置，显示为任务 ID；

Quantity： 该域名创建的任务总数；

Id： 每条任务 ID；

Status：执行状态 完成（Completed）、进行中（InProgress）、失败（Failed）

8 查询任务刷新详情信息 API

8.1 API 描述

根据任务 ID，查询该任务刷新的详细信息。

8.2 请求示例

8.2.1 语法

```
GET
/2015-09-17/distribution/{DistributionId}/invalidation/{Id}
```

8.2.2 请求参数

名称	描述	备注	必须
DistributionId	域名的 base64 编码		是
ID	任务 ID	可通过创建任务成功时返回的任务 ID 或通过查询刷新任务列表接口获得；	是

8.2.3 请求示例

```
GET
/2015-09-17/distribution/{DistributionId}/invalidation/{Id}
HTTP/1.0
Host: cdnapi.ksyun.com
Authorization: authentication string
```

```
Content-Type:application/xml
x-action: GetInvalidation
x-version: 2015-09-17
```

8.3 响应示例

```
<?xml version="1.0" encoding="UTF-8"?>
<Invalidation>
   <Id>3e4b66f8-cbf4-4ba6-95b0-3df3da24be28</Id>
   <Status>Completed</Status>
   <CreateTime>2016-04-05T13:42:37.000Z</CreateTime>
   <InvalidationBatch>
      <Paths>
         <Items>
            <Path>/wangweitest/info_01.html</Path>
            <Path>/wangweitest/</Path>
         </Items>
      </Paths>
      <CallerReference>10001</CallerReference>
   </InvalidationBatch>
</Invalidation>
```

注释：

CallerReference： 判断是否有下一页；

Id： 具体任务 ID， 可以根据接口 7：查询刷新任务列表 API 获得；

Status：执行状态 完成（Completed）、进行中（InProgress）、失败（Failed）

9 查询预加载任务列表 API

9.1 API 描述

对该用户下指定域名的全部预加载任务查询。结果根据时间倒序排序。

9.2 请求示例

9.2.1 语法

```
GET
/2015-09-
17/distribution/{DistributionId}/preload?Marker={Marker}&MaxItems={MaxItems}
```

9.2.2 请求参数

名称	描述	备注	必须
marker	指定任务查询的开始位置	输入要查询开始的任务 ID，ID 按倒序排序；	否
MaxItems	每一页显示的最大任务个	为空时默认 10	否

	数		
DistributionId	域名的 base64 编码		是

9.2.3 请求示例

```
GET
/2015-09-
17/distribution/{DistributionId}/preload?Marker={Marker}&MaxItems={MaxItems}
HTTP/1.0
Host: cdnapi.ksyun.com
Authorization: authentication string
Content-Type:application/xml
x-action: ListPreloads
x-version: 2015-09-17
```

9.3 响应示例

```
<?xml version="1.0" encoding="UTF-8"?>
<PreloadList>
   <NextMarker>360c87cf-597c-45df-a7ba-6aae96612373</NextMarker>
   <MaxItems>3</MaxItems>
   <IsTruncated>true</IsTruncated>
   <Quantity>16</Quantity>
   <Items>
      <PreloadSummary>
         <Id>3b2347d7-5d73-45cf-b476-f99ed00396c4</Id>
         <CreateTime>2016-04-05T15:02:19.000Z</CreateTime>
         <Status>Completed</Status>
      </PreloadSummary>
      <PreloadSummary>
         <Id>73145e21-112b-4919-b1a8-224177348108</Id>
         <CreateTime>2016-04-05T15:01:45.000Z</CreateTime>
         <Status>Completed</Status>
      </PreloadSummary>
      <PreloadSummary>
         <Id>6a285718-fa6a-45d0-b81f-65c7999d2dd8</Id>
         <CreateTime>2016-04-05T15:01:35.000Z</CreateTime>
         <Status>Completed</Status>
      </PreloadSummary>
   </Items>
</PreloadList>
```

注释：

IsTruncated： 判断是否有下一页；

NextMarker： 下一页开始的位置，显示为任务 ID；

Quantity： 该域名创建的任务总数；

Id： 每条任务 ID；

Status：执行状态 完成（Completed）、进行中（InProgress）、失败（Failed）

10 查询任务预加载详情信息 API

10.1 API 描述

根据任务 ID，查询该任务预加载的详细信息。

10.2 请求示例

10.2.1 语法

```
GET
/2015-09-17/distribution/{DistributionId}/preload/{Id}
```

10.2.2 请求参数

名称	描述	备注	必须
DistributionId	域名的 base64 编码		是
ID	任务 ID	可通过创建任务成功时返回的任务 ID 或通过查询预加载任务列表接口获得；	是

10.2.3 请求示例

```
GET
/2015-09-17/distribution/{DistributionId}/preload/{Id}HTTP/1.0
Host: cdnapi.ksyun.com
Authorization: authentication string
Content-Type:application/xml
x-action: GetPreload
x-version: 2015-09-17
```

10.3 响应示例

```
<?xml version="1.0" encoding="UTF-8"?>
<Preload>
   <Id>a00e7ad5-0223-47bb-bd85-56ffef5de590</Id>
   <Status>Completed</Status>
   <CreateTime>2016-04-05T15:01:01.000Z</CreateTime>
   <PreloadBatch>
      <Paths>
         <Items>
            <Path>/info_02.html</Path>
            <Path>/info_01.html</Path>
         </Items>
      </Paths>
```

```
      <CallerReference>10002</CallerReference>
   </PreloadBatch>
</Preload>
```

注释：

CallerReference： 判断是否有下一页；

Id： 具体任务 ID， 可以根据接口 9：查询预加载任务列表 API 获得；

Status：执行状态 完成（Completed）、进行中（InProgress）、失败（Failed）