开发指南

管控 API 参考

管控 API 参考

API 版本说明

本文介绍消息队列 Kafka 所有的 API 版本发布信息。

详情请见下表。

API 版本	发布日期	说明
V1.0.3	2018 年 12 月 6 日	- 新增根据实例 ID 创建 Topic 的接口: CreateTopic - 新增根据实例 ID 创建 ConsumerGroup 的接口 ：CreateConsumerGroup
V1.0.1	2018 年 10 月 25 日	GetConsumerProgress 接口的返回值数据结构 GetConsumerProgressResponse.ConsumerProgress.TopicListItem.OffsetListItem 增加属性分区字段“partition”
V1.0.0	2018 年 10 月 25 日	消息队列 Kafka 第一版 API 提供以下查询类接口： - 根据 AccessKeyId/AccessKeySecret 获取的实例列表: GetInstanceList - 根据实例 ID 获取实例下的 Topic 列表 ：GetTopicList - 根据 Topic 获取 Topic 信息 ：GetTopicStatus

		- 根据实例 ID 获取消费 ID 列表: GetConsumerList - 根据 Consumer ID 获取消费进度信息: GetConsumerProgress

接入指南

消息队列 Kafka 的管控 API 提供获取实例、Topic 和 Consumer Group 信息的接口。

本文介绍消息队列 Kafka SDK 的获取、初始参数的设置、使用方法概述等。

获取 SDK

以 Java 应用为例，在配置文件中添加依赖的 SDK 信息：

```
    <dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-alikafka</artifactId>
<version>1.0.5</version>
</dependency>
<dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-core</artifactId>
<version>4.3.9</version>
</dependency>
```

设置公共参数

构建并启动客户端时需要设置一系列参数信息，具体示例如下：

```
public static void main(String[] args) {
//构建 Client
IAcsClient iAcsClient = buildAcsClient();
}
private static IAcsClient buildAcsClient() {
//鉴权使用的 AccessKeyId，由阿里云官网控制台获取
String accessKey = "XXXXXX";
```

```
//鉴权使用的 AccessKeySecret，由阿里云官网控制台获取
String secretKey = "XXXXXX";

//产品 Code，消息队列 Kafka 产品常量值为 “alikafka”
String productName = "alikafka";

//API 的网关所在地域，目前支持的有 cn-beijing 和 cn-hangzhou 等
String regionId = "cn-beijing";
//接入点名称同 regionId 一致
String endPointName = "cn-beijing";
//对应 endPoint 的域名
String domain = "alikafka.cn-beijing.aliyuncs.com";

try {
DefaultProfile.addEndpoint(endPointName, regionId, productName, domain);
} catch (ClientException e) {
//log error
}
//构造 Client
IClientProfile profile = DefaultProfile.getProfile(regionId, accessKey, secretKey);
return new DefaultAcsClient(profile);
}
```

接入点列表

地域名称	RegionId	Domain
华东1（杭州）	cn-hangzhou	alikafka.cn-hangzhou.aliyuncs.com
华东2（上海）	cn-shanghai	alikafka.cn-shanghai.aliyuncs.com
华北1（青岛）	cn-qingdao	alikafka.cn-qingdao.aliyuncs.com
华北2（北京）	cn-beijing	alikafka.cn-beijing.aliyuncs.com
华北3（张家口）	cn-zhangjiakou	alikafka.cn-zhangjiakou.aliyuncs.com
华北5（呼和浩特）	cn-huhehaote	alikafka.cn-huhehaote.aliyuncs.com
华南1（深圳）	cn-shenzhen	alikafka.cn-shenzhen.aliyuncs.com
香港	cn-hongkong	alikafka.cn-hongkong.aliyuncs.com
新加坡	ap-southeast-1	alikafka.ap-southeast-1.aliyuncs.com

接口概览

以下是消息队列 Kafka 目前支持的管控 API 的概览说明。

实例管理接口

- GetInstanceList：查询实例信息列表

Topic 管理接口

- CreateTopic：创建 Topic
- GetTopicList：获取 Topic 列表
- GetTopicStatus：查询 Topic 消息收发信息

说明：更多接口将陆续开放。

Consumer Group 接口

- CreateConsumerGroup：创建 Consumer Group
- GetConsumerList：查询 Consumer Group 列表
- GetConsumerProgress：查询 Consumer Group 消费进度

说明：更多接口将陆续开放。

使用限制

单用户单接口的请求频率的限制为 3 QPS。更新和创建类接口的使用限制，请参见相应接口文档中的**使用限制**内容。

实例管理接口

GetInstanceList

使用 GetInstanceList 接口来获取您在某地域（Region）下所购买的实例的信息。

请求参数列表

名称	类型	是否必需	描述

RegionId	String	是	实例所属的地域

地域列表

地域名称	RegionId
华东1（杭州）	cn-hangzhou
华东2（上海）	cn-shanghai
华北1（青岛）	cn-qingdao
华北2（北京）	cn-beijing
华北3（张家口）	cn-zhangjiakou
华北5（呼和浩特）	cn-huhehaote
华南1（深圳）	cn-shenzhen
香港	cn-hongkong
新加坡	ap-southeast-1

返回参数列表

名称	类型	描述
RequestId	String	请求的唯一标识 ID
Code	Integer	返回码，返回 “200” 代表成功
Message	String	描述信息
InstanceList	Array	实例列表信息

InstanceList 数据结构列表

名称	类型	描述
InstanceId	String	实例 ID
RegionId	String	部署的实例所在地域的 RegionId
ServiceStatus	Integer	服务状态
VpcId	String	VPC 的 ID
VSwitchId	String	VSwitch 的 ID
EndPoint	String	接入点信息
CreateTime	Long	创建时间
ExpiredTime	Long	过期时间

使用示例

该示例为查询您在“华北2（北京）”地域下的实例列表。

```
public static void main(String[] args) {
//构建 Client
IAcsClient iAcsClient = buildAcsClient();

//构造获取实例信息 request
GetInstanceListRequest request = new GetInstanceListRequest();

//必要参数，注意此参数是指定获取哪个地域下的实例信息
request.setRegionId("cn-beijing");
request.setAcceptFormat(FormatType.JSON);

//获取返回值
try {
GetInstanceListResponse response = iAcsClient.getAcsResponse(request);
if (200 == response.getCode()) {
List<InstanceListItem> instanceListItems = response.getInstanceList();
if (CollectionUtils.isNotEmpty(instanceListItems)) {
for (InstanceListItem instance : instanceListItems) {
String instanceId = instance.getInstanceId();
System.out.println(instanceId);
//.....
}
}
} else {
//log warn
}
} catch (ClientException e) {
//log error
}

}
private static IAcsClient buildAcsClient() {
//产品 Code
String productName = "alikafka";

//您的 AccessKeyId 和 AccessKeySecret
String accessKey = "xxxxxx";
String secretKey = "xxxxxx";

//设置接入点相关参数，通常 regionId 值和 endPointName 值相等，接入点也是用对应地域的 domain
String regionId = "cn-beijing";
String endPointName = "cn-beijing";
String domain = "alikafka.cn-beijing.aliyuncs.com";
try {
DefaultProfile.addEndpoint(endPointName, regionId, productName, domain);
} catch (ClientException e) {
//log error
}

//构造 Client
```

```
IClientProfile profile = DefaultProfile.getProfile(regionId, accessKey, secretKey);
return new DefaultAcsClient(profile);
}
```

Topic 管理接口

CreateTopic

使用 CreateTopic 接口，根据实例 ID 创建 Topic。

使用限制

该接口有以下使用限制：

接口调用受白名单限制，请提交工单申请将用户 ID 加入白名单。

单用户请求频率限制为 1 QPS。

每个实例下最多可创建的 Topic 数量与您所购买的实例版本相关。详情请参见计费说明。

请求参数列表

名称	类型	是否必需	描述
InstanceId	String	是	实例 ID，可使用 GetInstanceList 获取
RegionId	String	是	此实例所在地域（Region），对应的 RegionId 请参见接入指南
Topic	String	是	创建的 Topic 名称，限制 64 个字符
Remark	String	是	创建的 Topic 的备注，限制 64个字符

返回参数列表

名称	类型	描述
RequestId	String	请求的唯一标识 ID
Code	Integer	返回码，返回 “200” 为成功
Success	Boolean	是否成功
Message	String	描述信息

使用示例

该示例为在“华北2（北京）”地域的某实例下创建一个 Topic。

```
public static void main(String[] args) {
//构建 Client
IAcsClient iAcsClient = buildAcsClient();
//构造创建 Topic 信息的 request
CreateTopicRequest request = new CreateTopicRequest();
request.setAcceptFormat(FormatType.JSON);
//必要参数，实例 ID
request.setInstanceId("alikafka_XXXXXXXXXXXXXX");
//必要参数，实例所在的地域
request.setRegionId("cn-beijing");

//必要参数 Topic 名称，限制在 64 个字符以内
request.setTopic("alikafka_for_test_1");
//必要参数 Topic 备注，限制在 64 个字符以内
request.setRemark("myTest");

//获取返回值
try {
CreateTopicResponse response = iAcsClient.getAcsResponse(request);
//log requestId for trace
String requestId = response.getRequestId();
if (200 == response.getCode()) {
if (response.getSuccess()) {
//log success
} else {
//log warn
}
} else {
//log warn
}
} catch (ClientException e) {
//log error
}
}
private static IAcsClient buildAcsClient() {
//产品 Code
String productName = "alikafka";
```

```
//您的 AccessKeyId 和 AccessKeySecret
String accessKey = "xxxxxx";
String secretKey = "xxxxxx";

//设置接入点相关参数，通常 regionId 值和 endPointName 值相等，接入点也是用对应地域的 domain
String regionId = "cn-beijing";
String endPointName = "cn-beijing";
String domain = "alikafka.cn-beijing.aliyuncs.com";
try {
DefaultProfile.addEndpoint(endPointName, regionId, productName, domain);
} catch (ClientException e) {
//log error
}

//构造 Client
IClientProfile profile = DefaultProfile.getProfile(regionId, accessKey, secretKey);
return new DefaultAcsClient(profile);
}
```

GetTopicList

使用 GetTopicList 接口，根据实例 ID 获取该实例下的 Topic 列表。

请求参数列表

名称	类型	是否必需	描述
InstanceId	String	是	实例 ID，可使用 GetInstanceList 获取
RegionId	String	是	此实例所在地域（Region），对应的 RegionId 请参见接入指南

返回参数列表

名称	类型	描述
RequestId	String	请求的唯一标识 ID
Code	Integer	返回码，返回 “200” 为成功
Message	String	描述信息
TopicList	Array	实例列表信息

TopicList 数据结构列表

名称	类型	描述
Topic	String	Topic 名称
CreateTime	Long	Topic 创建时间
Remark	String	Topic 备注
Status	Integer	状态码，取值说明如下: - “0” 代表服务中； - “1” 代表冻结； - “2” 代表暂停。
InstanceId	String	实例 ID
RegionId	String	实例所在地域的 ID
StatusName	String	状态描述

使用示例

该示例为在“华北2（北京）”地域，查询某个实例下的 Topic 列表。

```
public static void main(String[] args) {
//构建 Client
IAcsClient iAcsClient = buildAcsClient();

//构造获取 topicList 信息的 request
GetTopicListRequest request = new GetTopicListRequest();
request.setAcceptFormat(FormatType.JSON);

//必要参数实例 ID
request.setInstanceId("alikafka_pre-xxxxxxxxxxxx");
//必要参数，此实例所在地域，必须使用 GetInstanceList 返回值的实例所在的地域
request.setRegionId("cn-beijing");

//获取返回值
try {
GetTopicListResponse response = iAcsClient.getAcsResponse(request);
if (200 == response.getCode()) {
List<GetTopicListResponse.TopicListItem> topicListItems = response.getTopicList();
if (CollectionUtils.isNotEmpty(topicListItems)) {
for (TopicListItem item : topicListItems) {
String topic = item.getTopic();
System.out.println(topic);
//.....
}
}
} else {
//log warn
```

```
}
} catch (ClientException e) {
//log error
}
}
private static IAcsClient buildAcsClient() {
//产品 Code
String productName = "alikafka";

//您的 AccessKeyId 和 AccessKeySecret
String accessKey = "xxxxxx";
String secretKey = "xxxxxx";

//设置接入点相关参数，通常 regionId 值和 endPointName 值相等，接入点也是用对应地域的 domain
String regionId = "cn-beijing";
String endPointName = "cn-beijing";
String domain = "alikafka.cn-beijing.aliyuncs.com";
try {
DefaultProfile.addEndpoint(endPointName, regionId, productName, domain);
} catch (ClientException e) {
//log error
}

//构造 Client
IClientProfile profile = DefaultProfile.getProfile(regionId, accessKey, secretKey);
return new DefaultAcsClient(profile);
}
```

GetTopicStatus

使用 GetTopicStatus 接口，根据实例 ID 和 Topic 名称来查询某 Topic 的消息收发数据。

请求参数列表

名称	类型	是否必需	描述
RegionId	String	是	此实例所在地域（Region），对应的 RegionId 请参见接入指南
InstanceId	String	是	实例 ID，可使用 GetInstanceList 获取
Topic	String	是	Topic 名称，可使用 GetTopicList 获取

返回参数列表

名称	类型	描述
RequestId	String	请求唯一标识 ID
Code	Integer	返回码，返回 “200” 代表成功
Message	String	描述信息
TopicStatus	Struct	TopicStatus 的数据结构

TopicStatus 的数据结构列表

名称	类型	描述
TotalCount	Long	消息总数
LastTimeStamp	Long	最后被消费的一条消息的产生的时间
OffsetTable	Array	Offset 列表

Offset 的数据结构列表

名称	类型	描述
Topic	String	Topic 名称
Partition	Integer	分区的 ID
MinOffset	Long	消息最小位点
MaxOffset	Long	消息最大位点
LastUpdateTimestamp	Long	最后被消费的一条消息的产生时间

使用示例

该示例为在“华北2（北京）”地域，查询某实例下的某个 Topic 的消息收发信息。

```
public static void main(String[] args) {
//构建 Client
IAcsClient iAcsClient = buildAcsClient();

//构造获取 Topic 信息的 request
GetTopicStatusRequest request = new GetTopicStatusRequest();
request.setAcceptFormat(FormatType.JSON);
```

```
//必要参数，实例 ID
request.setInstanceId("alikafka_pre-xxxxxx");
//必要参数，此实例所在地域，必须使用 GetInstanceList 返回值的实例所在的地域
request.setRegionId("cn-beijing");
//必要参数，Topic 名称
request.setTopic("test-xxxxxx");

//获取返回值
try {
GetTopicStatusResponse response = iAcsClient.getAcsResponse(request);
if (200 == response.getCode()) {
GetTopicStatusResponse.TopicStatus topicStatus = response.getTopicStatus();
if (topicStatus != null) {
Long totalCount = topicStatus.getTotalCount();
System.out.println(totalCount);
for (GetTopicStatusResponse.TopicStatus.OffsetTableItem item : topicStatus.getOffsetTable()) {
item.getTopic();
item.getPartition();
item.getMaxOffset();
//.....
}
}
} else {
//log warn
}
} catch (ClientException e) {
//log error
}
}
private static IAcsClient buildAcsClient() {
//产品 Code
String productName = "alikafka";

//您的 AccessKeyId 和 AccessKeySecret
String accessKey = "xxxxxx";
String secretKey = "xxxxxx";

//设置接入点相关参数，通常 regionId 值和 endPointName 值相等，接入点也是用对应地域的 domain
String regionId = "cn-beijing";
String endPointName = "cn-beijing";
String domain = "alikafka.cn-beijing.aliyuncs.com";
try {
DefaultProfile.addEndpoint(endPointName, regionId, productName, domain);
} catch (ClientException e) {
//log error
}

//构造 Client
IClientProfile profile = DefaultProfile.getProfile(regionId, accessKey, secretKey);
return new DefaultAcsClient(profile);
}
```

Consumer Group 管理接口

CreateConsumerGroup

使用 CreateConsumerGroup 接口，根据实例 ID 创建 Consumer Group。

使用限制

该接口有以下使用限制：

接口调用受白名单限制，请提交工单申请将用户 ID 加入白名单。

单用户请求频率限制为 1 QPS。

每个实例下最多可创建 100 个 Consumer Group。

请求参数列表

名称	类型	是否必需	描述
InstanceId	String	是	实例 ID，可使用 GetInstanceList 获取
RegionId	String	是	此实例所在地域（Region），对应的 RegionId 请参见接入指南
ConsumerId	String	是	创建的 Consumer Group ID，限制 64 个字符

返回参数列表

名称	类型	描述
RequestId	String	请求的唯一标识 ID
Code	Integer	返回码，返回 “200” 为成功
Success	Boolean	成功与否

Message	String	描述信息

使用示例

该示例为在“华北2（北京）”地域，在某个实例下创建一个 Consumer Group。

```
public static void main(String[] args) {
//构建 Client
IAcsClient iAcsClient = buildAcsClient();
//构造创建 Consumer Group 的 request
CreateConsumerGroupRequest request = new CreateConsumerGroupRequest();
request.setAcceptFormat(FormatType.JSON);
//必要参数，实例 ID
request.setInstanceId("alikafka_XXXXXXXXXXXXX");
//必要参数，实例所在地域
request.setRegionId("cn-beijing");

//必要参数，Consumer Group ID，限制在 64 个字符以内
request.setConsumerId("alikafka_for_test_XXXXXX");

//获取返回值
try {
CreateConsumerGroupResponse response = iAcsClient.getAcsResponse(request);
//log requestId for trace
String requestId = response.getRequestId();
if (200 == response.getCode()) {
if (response.getSuccess()) {
//log success
} else {
//log warn
}
} else {
//log warn
}
} catch (ClientException e) {
//log error
}
}
private static IAcsClient buildAcsClient() {
//产品 Code
String productName = "alikafka";

//您的 AccessKeyId 和 AccessKeySecret
String accessKey = "xxxxxx";
String secretKey = "xxxxxx";

//设置接入点相关参数，通常 regionId 值和 endPointName 值相等，接入点也是用对应地域的 domain
String regionId = "cn-beijing";
String endPointName = "cn-beijing";
String domain = "alikafka.cn-beijing.aliyuncs.com";
try {
DefaultProfile.addEndpoint(endPointName, regionId, productName, domain);
} catch (ClientException e) {
//log error
```

```
}

//构造 Client
IClientProfile profile = DefaultProfile.getProfile(regionId, accessKey, secretKey);
return new DefaultAcsClient(profile);
}
```

GetConsumerList

使用 GetConsumerList 接口，根据实例 ID 获取该实例下的 Consumer Group 列表。

请求参数列表

名称	类型	是否必需	描述
InstanceId	String	是	实例 ID，可使用 GetInstanceList 获取
RegionId	String	是	此实例所在地域，对应的 RegionId 请参见接入指南

返回参数列表

名称	类型	描述
RequestId	String	请求的唯一标识 ID
Code	Integer	返回码，返回 “200” 为成功
Message	String	描述信息
ConsumerList	Array	Consumer Group 列表信息

ConsumerList 数据结构列表

名称	类型	描述
RegionId	String	地域的 ID
InstanceId	Long	实例 ID
ConsumerId	String	Consumer Group ID

使用示例

该示例为在“华北2（北京）”地域，查询某个实例 ID 下的 Consumer Group 列表。

```
public static void main(String[] args) {
//构建 Client
IAcsClient iAcsClient = buildAcsClient();

//构造获取 consumerList 信息 request
GetConsumerListRequest request = new GetConsumerListRequest();
request.setAcceptFormat(FormatType.JSON);

//必要参数实例 ID
request.setInstanceId("alikafka_pre-xxxxxx");
//必要参数，此实例所在地域，必须使用 GetInstanceList 返回值的实例所在的地域
request.setRegionId("cn-beijing");

//获取返回值
try {
GetConsumerListResponse response = iAcsClient.getAcsResponse(request);
if (200 == response.getCode()) {
List<ConsumerListItem> consumerList = response.getConsumerList();
if (CollectionUtils.isNotEmpty(consumerList)) {
for (ConsumerListItem item : consumerList) {
String consumerId = item.getConsumerId();
System.out.println(consumerId);
//.....
}
}
} else {
//log warn
}
} catch (ClientException e) {
//log error
}
}
private static IAcsClient buildAcsClient() {
//产品 Code
String productName = "alikafka";

//您的 AccessKeyId 和 AccessKeySecret
String accessKey = "xxxxxx";
String secretKey = "xxxxxx";

//设置接入点相关参数，通常 regionId 值和 endPointName 值相等，接入点也是用对应地域的 domain
String regionId = "cn-beijing";
String endPointName = "cn-beijing";
String domain = "alikafka.cn-beijing.aliyuncs.com";
try {
DefaultProfile.addEndpoint(endPointName, regionId, productName, domain);
} catch (ClientException e) {
//log error
}
```

```
//构造 Client
IClientProfile profile = DefaultProfile.getProfile(regionId, accessKey, secretKey);
return new DefaultAcsClient(profile);
}
```

GetConsumerProgress

使用 GetConsumerProgress 接口，根据实例 ID 和 Consumer Group ID 查询该 Consumer Group 的消费进度信息。

请求参数列表

名称	类型	是否必需	描述
RegionId	String	是	此实例所在地域，对应的 RegionId 请参见接入指南
InstanceId	String	是	实例 ID，可使用 GetInstanceList 获取
ConsumerId	String	是	Consumer Group ID，可使用 GetConsumerList 获取

返回参数列表

名称	类型	描述
RequestId	String	请求的唯一标识 ID
Code	Integer	返回码，返回 “200” 为成功
Message	String	描述信息
ConsumerProgress	Struct	ConsumerProgress 的数据结构

ConsumerProgress 的数据结构列表

名称	类型	描述
TotalDiff	Long	此 Consumer Group 未消费的消息总量，即堆积量

LastTimestamp	Long	此 Consumer Group 最后被消费的消息的产生时间
TopicList	Array	此 Consumer Group 对应的每个 Topic 的消费进度列表

TopicList 的数据结构列表

名称	类型	描述
Topic	String	Topic 名称
TotalDiff	Integer	该 Topic 的未消费消息总量，即堆积量
LastTimestamp	Long	该 Topic 最后被消费的消息的产生时间
OffsetList	Array	该 Topic 的消费位点信息

OffsetList 的数据结构列表

名称	类型	描述
Partition	Integer	分区的 ID
BrokerOffset	Long	最大位点
ConsumerOffset	Long	消费位点
LastTimestamp	Long	该分区最后被消费的消息的产生时间

使用示例

该示例为在“华北2（北京）”地域，查询某个实例下的某个 Consumer Group 的消费进度。

```
public static void main(String[] args) {
//构建 Client
IAcsClient iAcsClient = buildAcsClient();

//构造获取 Consumer Group 消费进度的 request
GetConsumerProgressRequest request = new GetConsumerProgressRequest();
request.setAcceptFormat(FormatType.JSON);

//必要参数，此实例所在地域，必须使用 GetInstanceList 返回值的实例所在的地域
request.setRegionId("cn-beijing");
//必要参数，实例 ID
request.setInstanceId("alikafka_pre-xxxxxx");
//必要参数，Consumer Group ID
request.setConsumerId("CID_xxxxxx");
```

```
//获取返回值
try {
GetConsumerProgressResponse response = iAcsClient.getAcsResponse(request);
if (200 == response.getCode()) {
GetConsumerProgressResponse.ConsumerProgress consumerProgress = response.getConsumerProgress();
long totalDiff = consumerProgress.getTotalDiff();
System.out.println(totalDiff);
for (GetConsumerProgressResponse.ConsumerProgress.TopicListItem item : consumerProgress.getTopicList()) {
System.out.println(item.getTopic());;
item.getTotalDiff();
List<OffsetListItem> offsetListItems = item.getOffsetList();
for (OffsetListItem offsetListItem : offsetListItems) {
long brokerOffset = offsetListItem.getBrokerOffset();
long consumerOffset = offsetListItem.getConsumerOffset();
System.out.println("brokerOffset="+brokerOffset);
System.out.println("consumerOffset="+consumerOffset);
//.....
}
}
} else {
//log warn
}
} catch (ClientException e) {
//log error
}
}
private static IAcsClient buildAcsClient() {
//产品 Code
String productName = "alikafka";

//您的 AccessKeyId 和 AccessKeySecret
String accessKey = "xxxxxx";
String secretKey = "xxxxxx";

//设置接入点相关参数，通常 regionId 值和 endPointName 值相等，接入点也是用对应地域的 domain
String regionId = "cn-beijing";
String endPointName = "cn-beijing";
String domain = "alikafka.cn-beijing.aliyuncs.com";
try {
DefaultProfile.addEndpoint(endPointName, regionId, productName, domain);
} catch (ClientException e) {
//log error
}

//构造 Client
IClientProfile profile = DefaultProfile.getProfile(regionId, accessKey, secretKey);
return new DefaultAcsClient(profile);
}
```

HTTP 调用方式

概述

本文适用基于 API URL 发起 HTTP/HTTPS POST 请求的用户，如果您使用的是 SDK、阿里云 CLI 或者 API Explorer，可以跳过此环节。

发起 API 请求的 URL 由不同参数拼凑而成，有固定的请求结构。URL 中包含公共参数、您的签名和某个 API 的具体参数。每篇 API 文档均给出了 URL 请求示例供您参考，但是为了方便显示，我们并没有编码这些 URL 示例，您需要在发起请求前自行编码。我们根据您的签名验证了请求后，会返回结果给您。接口调用成功会显示返回参数，调用失败则显示相应报错，您可以根据公共错误码和具体 API 错误码分析排查。返回参数的详情请参见 SDK 接口文档。

说明：推荐您使用 SDK，以免除您手动签名验证环节，方便调用接口以及管理资源。

公共参数

公共参数分为公共请求参数和公共返回参数。

公共请求参数

以下公共请求参数适用于通过 URL 发送 POST 请求调用消息队列 Kafka API。

名称	类型	是否必需	描述
Action	String	是	API 的名称。取值请参见接入指南。
AccessKeyId	String	是	访问密钥 ID。AccessKey（包含 AccessKeyId 和 AccessKeySecret）用于调用 API，而用户密码用于登录消息队列 Kafka 控制台。
Signature	String	是	您的签名。取值请参见签名机制。
SignatureMethod	String	是	签名方式。取值范围：HMAC-SHA1。
SignatureVersion	String	是	签名算法版本。取值范围：1.0。

SignatureNonce	String	是	签名唯一随机数。用于防止网络重放攻击，建议您每一次请求都使用不同的随机数。
Timestamp	String	是	请求的时间戳。按照 ISO8601 标准表示，并需要使用 UTC 时间，格式为 yyyy-MM-ddTHH:mm:ssZ。示例：2018-01-01T12:00:00Z 表示北京时间 2018 年 1 月 1 日 20 点 00 分 00 秒。
Version	String	是	API 的版本号，格式为 YYYY-MM-DD。取值范围：2018-10-15。
Format	String	否	返回参数的语言类型。取值范围：json 或 xml。默认值：xml。

请求示例

```
https:/alikafka.cn-hangzhou.aliyuncs.com/?Action=XXXXXX
&Format=xml
&Version=2014-05-26
&Signature=Pc5WB8gokVn0xfeu%2FZV%2BiNM1dgI%3D
&SignatureMethod=HMAC-SHA1
&SignatureNonce=15215528852396
&SignatureVersion=1.0
&AccessKeyId=key-test
&Timestamp=2018-01-01T12:00:00Z
...
```

公共返回参数

名称	类型	描述
RequestId	String	请求 ID。无论调用接口成功与否，我们都会返回请求 ID。

请求示例

本文提供使用 Java 语言构造参数和签名并 POST 发送请求的示例，其他语言流程类似。

使用示例

```
  private static final String ISO8601_DATE_FORMAT = "yyyy-MM-dd'T'HH:mm:ss'Z'";
private static final String ENCODING = "UTF-8";
private static final String HTTP_METHOD = "POST";
//购买的实例所在地域的 Region ID
private static final String REGION_ID = "cn-xxxxxx";
private static final String ACCESS_KEY = "xxxxxx";
private static final String ACCESS_KEY_SECRET = "xxxxxx";


public static void main(String[] args) {
try {
// 1. 设置参数
Map<String, String> parameters = buildCommonParams();
// 2. 排序请求参数: 根据字母排序
String[] sortedKeys = sortParamsToArray(parameters);
// 3. 构造 stringToSign 字符串
String stringToSign = buildSignString(parameters, sortedKeys);
// 4. 计算签名
String signature = calculateSignature(stringToSign);
// 5. 将签名设置到参数中
parameters.put("Signature", signature);
// 6. 发送 POST 请求
String result = post(String.format("http://alikafka.%s.aliyuncs.com/", REGION_ID), parameters);
// 7. 验证结果
System.out.println(result);
} catch (Throwable throwable) {
throwable.printStackTrace();
}
}

private static Map<String, String> buildCommonParams() {
Map<String, String> parameters = Maps.newHashMap();
// Action 为请求方法
// GetInstanceList：获取实例信息；GetConsumerList：获取消费组列表；GetTopicList：获取 Topic 列表
；GetTopicStatus：获取 Topic 状态
// GetConsumerProgress：获取消费状态；CreateTopic：创建 Topic；CreateConsumerGroup：创建 Consumer Group
parameters.put("Action", "GetInstanceList");
// 接口版本："2018-10-15"
parameters.put("Version", "2018-10-15");
// 请求参数：RegionId 为必填参数
// 如果接口有其他参数请都设置：比如 InstanceId/Topic/ConsumerId
// parameters.put("InstanceId", "cn-huhehaote");
// parameters.put("Topic", "cn-huhehaote");
// parameters.put("Remark", "cn-huhehaote");
// parameters.put("ConsumerId", "cn-huhehaote");
parameters.put("RegionId", REGION_ID);
parameters.put("AccessKeyId", ACCESS_KEY);
```

```
// 时间戳，注意格式:yyyy-MM-dd'T'HH:mm:ss'Z'
parameters.put("Timestamp", formatIso8601Date(new Date()));
parameters.put("SignatureMethod", "HMAC-SHA1");
parameters.put("SignatureVersion", "1.0");
parameters.put("SignatureNonce", UUID.randomUUID().toString());
parameters.put("Format", "json");
return parameters;
}

private static String[] sortParamsToArray(Map<String, String> parameters) {
String[] sortedKeys = parameters.keySet().toArray(new String[]{});
Arrays.sort(sortedKeys);
return sortedKeys;
}

private static String buildSignString(Map<String, String> parameters,
String[] sortedKeys) throws UnsupportedEncodingException {
StringBuilder stringToSign = new StringBuilder();
String SEPARATOR = "&";
stringToSign.append(HTTP_METHOD).append(SEPARATOR);
stringToSign.append(percentEncode("/")).append(SEPARATOR);
StringBuilder canonicalizedQueryString = new StringBuilder();
for(String key : sortedKeys) {
// 这里注意编码 key 和 value
canonicalizedQueryString.append("&")
.append(percentEncode(key)).append("=")
.append(percentEncode(parameters.get(key)));
}

// 这里注意编码 canonicalizedQueryString
stringToSign.append(percentEncode(canonicalizedQueryString.toString().substring(1)));
return stringToSign.toString();
}

private static String calculateSignature(String stringToSign)
throws NoSuchAlgorithmException, InvalidKeyException, UnsupportedEncodingException {
String ALGORITHM = "HmacSHA1";
String ENCODING = "UTF-8";
//对应账号的 AccessKeySecret
String accessKeySecret = ACCESS_KEY_SECRET + "&";
Mac mac = Mac.getInstance(ALGORITHM);
mac.init(new SecretKeySpec(accessKeySecret.getBytes(ENCODING), ALGORITHM));
byte[] signData = mac.doFinal(stringToSign.getBytes(ENCODING));
return new String(Base64.getEncoder().encode(signData));
}

private static String percentEncode(String value) throws UnsupportedEncodingException {
return value != null ? URLEncoder.encode(value, ENCODING).replace("+", "%20").replace("*", "%2A").replace("%7E",
"~") : null;
}

private static String formatIso8601Date(Date date) {
SimpleDateFormat df = new SimpleDateFormat(ISO8601_DATE_FORMAT);
df.setTimeZone(new SimpleTimeZone(0, "GMT"));
return df.format(date);
}
```

```
private static String post(String url, Map<String, String> paramMap) throws IOException {
Form form = Form.form();
for (String key : paramMap.keySet()) {
form.add(key, paramMap.get(key));
}

return Request.Post(url).bodyForm(form.build()).connectTimeout(10000).execute().returnContent().asString();
}
```

请求结构

消息队列 Kafka API 支持基于 URL 发起 HTTP/HTTPS POST 请求。请求参数需要包含在 URL 中。本文列举了 POST 请求中的结构解释，并提供了消息队列 Kafka 的服务接入地址（Endpoint）。

结构示例

以下为 GetInstanceList 一条未编码的 URL 请求示例：

```
http://alikafka.cn-hangzhou.aliyuncs.com/?Action=GetInstanceList
&RegionId=cn-hangzhou
&<公共请求参数>
```

http 指定了请求通信协议。

alikafka.cn-hangzhou.aliyuncs.com 指定了消息队列 Kafka 的服务接入地址（Endpoint）。

Action=GetInstanceList 指定了要调用的 API。

<公共请求参数>是系统规定的公共参数。

通信协议

支持 HTTP 或 HTTPS协议请求通信。为了获得更高的安全性，推荐您使用 HTTPS 协议发送请求。

接入地址

地域名称	RegionId	Domain

华东1（杭州）	cn-hangzhou	alikafka.cn-hangzhou.aliyuncs.com
华东2（上海）	cn-shanghai	alikafka.cn-shanghai.aliyuncs.com
华北1（青岛）	cn-qingdao	alikafka.cn-qingdao.aliyuncs.com
华北2（北京）	cn-beijing	alikafka.cn-beijing.aliyuncs.com
华北3（张家口）	cn-zhangjiakou	alikafka.cn-zhangjiakou.aliyuncs.com
华北5（呼和浩特）	cn-huhehaote	alikafka.cn-huhehaote.aliyuncs.com
华南1（深圳）	cn-shenzhen	alikafka.cn-shenzhen.aliyuncs.com
香港	cn-hongkong	alikafka.cn-hongkong.aliyuncs.com
新加坡	ap-southeast-1	alikafka.ap-southeast-1.aliyuncs.com

请求参数

您需要通过 Action 参数指定目标操作，例如 Action=GetInstanceList，还需要指定接口的其他参数以及公共参数。Action 参数的可取值如下：

- GetInstanceList：获取实例信息；
- GetConsumerList：获取消费组列表；
- GetTopicList：获取 Topic 列表；
- GetTopicStatus：获取 Topic 状态；
- GetConsumerProgress：获取消费状态；
- CreateTopic：创建 Topic；
- CreateConsumerGroup：创建消费组。

字符编码

请求及返回结果都使用 UTF-8 字符集编码。

签名机制

对于每一次 HTTP 或者 HTTPS 协议请求，我们会根据访问中的签名信息验证访问请求者身份。具体使用 AccessKey 的 AccessKeyId 和 AccessKeySecret 对称加密验证实现。

您的用户名/密码是用于登录消息队列 Kafka 控制台的身份凭据，与此类似，AccessKey 是用于调用 API 的身份凭据。AccessKey 中的 AccessKeyId 是访问者身份，AccessKeySecret 是加密签名字符串和服务器端验证签名字符串的密钥，必须严格保密谨防泄露。

说明：消息队列 Kafka 提供了多种编程语言的 SDK，可以免去您签名的烦恼。更多详情，请下载 SDK。

步骤一：构造规范化请求字符串

排序参数。排序规则以首字母顺序排序，排序参数包括公共请求参数和接口自定义参数，不包括公共请求参数中的 Signature 参数。

编码参数。使用 UTF-8 字符集按照 RFC3986 规则编码请求参数和参数取值，编码规则如下：

字符 A~Z、a~z、0~9 以及字符 “-”、“_”、“.”、“~” 不编码。

其它字符编码成 %XY 的格式，其中 XY 是字符对应 ASCII 码的 16 进制。示例：半角双引号（” ）对应 %22。

扩展的 UTF-8 字符，编码成 %XY%ZA... 的格式。

空格（ ）编码成 %20，而不是加号（+）。

该编码方式与 application/x-www-form-urlencoded MIME 格式编码算法相似，但又有所不同。

如果您使用的是 Java 标准库中的 java.net.URLEncoder，可以先用标准库中 percentEncode 编码，随后将编码后的字符中加号（+）替换为 %20、星号（*）替换为 %2A、%7E 替换为波浪号（~），即可得到上述规则描述的编码字符串。示例如下：

```
private static final String ENCODING = "UTF-8";
private static String percentEncode(String value) throws UnsupportedEncodingException {
return value != null ? URLEncoder.encode(value, ENCODING).replace("+", "%20").replace("*",
"%2A").replace("%7E", "~") : null;
}
```

使用等号（=）连接编码后的请求参数和参数取值。

使用与号（&）连接编码后的请求参数，注意参数排序与步骤 1 一致。

现在，您得到了规范化请求字符串（CanonicalizedQueryString），其结构遵循请求结构。

步骤二：构造签名字符串

构造待签名字符串 StringToSign。您可以同样使用 percentEncode 处理上一步构造的规范化请求字符串，规则如下：

```
StringToSign=
HTTPMethod + "&" + //HTTPMethod：发送请求的 HTTP 方法，例如 POST。
percentEncode("/") + "&" + //percentEncode("/")：字符（/）UTF-8 编码得到的值，即 %2F。
percentEncode(CanonicalizedQueryString) //您的规范化请求字符串。
```

按照 RFC2104 的定义，计算待签名字符串 StringToSign 的 HMAC-SHA1 值。示例使用的是 Java Base64 编码方法。

```
Signature = Base64( HMAC-SHA1( AccessSecret, UTF-8-Encoding-Of(StringToSign) ) )
```

说明：计算签名时，RFC2104 规定的 Key 值是您的 AccessKeySecret 并加上与号（ &），其 ASCII 值为 38。

添加根据 RFC3986 规则编码后的参数 Signature 到规范化请求字符串URL中。

示例一：参数拼接法

以调用 GetInstanceList 获取实例列表为例。假设您获得了 AccessKeyID=testid 以及 AccessKeySecret=testsecret，签名流程如下所示：

构造规范化请求字符串。

```
http://alikafka.%s.aliyuncs.com/?Timestamp=2016-02-
23T12:46:24Z&Format=XML&AccessKeyId=testid&Action=GetInstanceList&SignatureMethod=HMAC-
SHA1&SignatureNonce=3ee8c1b8-83d3-44af-a94f-4e0ad82fd6cf&Version=2014-05-
26&SignatureVersion=1.0
```

构造待签名字符串 StringToSign。

```
POST&%2F&AccessKeyId%3Dtestid&Action%3DGetInstanceList&Format%3DXML&SignatureMethod%3D
HMAC-SHA1&SignatureNonce%3D3ee8c1b8-83d3-44af-a94f-
```

```
4e0ad82fd6cf&SignatureVersion%3D1.0&Timestamp%3D2016-02-
23T12%253A46%253A24Z&Version%3D2014-05-26
```

计算签名值。因为 AccessKeySecret=testsecret，用于计算的 Key 为 testsecret&，计算得到的签名值为 OLeaidS1JvxuMvnyHOwuJ+uX5qY=。示例使用的是 Java Base64 编码方法。

```
Signature = Base64( HMAC-SHA1( AccessSecret, UTF-8-Encoding-Of(StringToSign) ) )
```

添加 RFC3986 规则编码后的 Signature=OLeaidS1JvxuMvnyHOwuJ%2BuX5qY%3D 到步骤 1 的 URL 中。

```
http://alikafka.%s.aliyuncs.com/?SignatureVersion=1.0&Action=GetInstanceList&Format=JSON&SignatureNonce=3ee8c1b8-83d3-44af-a94f-4e0ad82fd6cf&Version=2014-05-26&AccessKeyId=testid&Signature=OLeaidS1JvxuMvnyHOwuJ%2BuX5qY%3D&SignatureMethod=HMAC-SHA1&Timestamp=2016-02-23T12%253A46%253A24Z
```

通过以上 URL，您可以使用浏览器、curl 或者 wget 等工具发起 HTTP 请求调用 GetInstanceList，查看阿里云的地域列表。

示例二：编程语言法

依然以调用 GetInstanceList 获取实例列表为例。假设您获得了 AccessKeyID=testid 以及 AccessKeySecret=testsecret，并且假定所有请求参数放在一个 Java Map<String, String> 对象里。

预定义编码方法。

```
private static final String ENCODING = "UTF-8";
private static String percentEncode(String value) throws UnsupportedEncodingException {
return value != null ? URLEncoder.encode(value, ENCODING).replace("+", "%20").replace("*",
"%2A").replace("%7E", "~") : null;
}
```

预定义编码时间格式 Timestamp。参数 Timestamp 必须符合 ISO8601 规范，并需要使用 UTC 时间，时区为 +0。

```
private static final String ISO8601_DATE_FORMAT = "yyyy-MM-dd'T'HH:mm:ss'Z'";
private static String formatIso8601Date(Date date) {
SimpleDateFormat df = new SimpleDateFormat(ISO8601_DATE_FORMAT);
df.setTimeZone(new SimpleTimeZone(0, "GMT"));
return df.format(date);
}
```

构造请求字符串。

```
final String HTTP_METHOD = "POST";
Map parameters = new HashMap();
// 输入请求参数
parameters.put("Action", "GetInstanceList");
parameters.put("Version", "2014-05-26");
parameters.put("AccessKeyId", "testid");
parameters.put("Timestamp", formatIso8601Date(new Date()));
parameters.put("SignatureMethod", "HMAC-SHA1");
parameters.put("SignatureVersion", "1.0");
parameters.put("SignatureNonce", UUID.randomUUID().toString());
parameters.put("Format", "JSON");
// 排序请求参数
String[] sortedKeys = parameters.keySet().toArray(new String[]{});
Arrays.sort(sortedKeys);
final String SEPARATOR = "&";
// 构造 stringToSign 字符串
StringBuilder stringToSign = new StringBuilder();
stringToSign.append(HTTP_METHOD).append(SEPARATOR);
stringToSign.append(percentEncode("/")).append(SEPARATOR);
StringBuilder canonicalizedQueryString = new StringBuilder();
for(String key : sortedKeys) {
// 这里注意编码 key 和 value
canonicalizedQueryString.append("&")
.append(percentEncode(key)).append("=")
.append(percentEncode(parameters.get(key)));
}
// 这里注意编码 canonicalizedQueryString
stringToSign.append(percentEncode(
canonicalizedQueryString.toString().substring(1)));
```

签名。因为 AccessKeySecret=testsecret，所以用于计算 HMAC 的 Key 为 testsecret&，计算得到的签名值为 OLeaidS1JvxuMvnyHOwuJ%2BuX5qY%3D。

```
// 以下是一段计算签名的示例代码
final String ALGORITHM = "HmacSHA1";
final String ENCODING = "UTF-8";
key = "testsecret&";
Mac mac = Mac.getInstance(ALGORITHM);
mac.init(new SecretKeySpec(key.getBytes(ENCODING), ALGORITHM));
byte[] signData = mac.doFinal(stringToSign.getBytes(ENCODING));
String signature = new String(Base64.encodeBase64(signData));
```