**SmartGo-Broker**

**技术文档说明**

针对版本V1.0.0

**©成都基础平台架构**

**2017/11/21**

Smargo-broker修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 修订内容 | 作者 | 审核 | 修订日期 |
| V1.0.0 | 初始版本 | 郜焱磊 、戎志宏 | 基础平台架构组 | 2017/11/21 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1 概述 4](#_Toc499219302)

[2 Borker模块交互 4](#_Toc499219303)

[2.1 Registry 4](#_Toc499219304)

[2.2 Client 4](#_Toc499219305)

[2.3 Net 4](#_Toc499219306)

[2.4 Store 5](#_Toc499219307)

[2.5 Broker 5](#_Toc499219308)

[3 专业术语 5](#_Toc499219309)

[3.1 Topic 5](#_Toc499219310)

[3.1 ConsumerOffset 5](#_Toc499219311)

[3.2 SubscriptionGroup 5](#_Toc499219312)

[4 Broker实现原理 5](#_Toc499219313)

[4.1 Topic 管理 5](#_Toc499219314)

[4.2 ConsumerOffset管理 7](#_Toc499219315)

[4.3 SubscriptionGroup管理 9](#_Toc499219316)

[4.4 发送消息 10](#_Toc499219317)

[4.5 消费消息 11](#_Toc499219318)

[4.6 主从同步 12](#_Toc499219319)

[4.7 Hold 13](#_Toc499219320)

[4.8 消息统计 14](#_Toc499219321)

[4.9 Producer、Consumer连接管理 15](#_Toc499219322)

[附件一 Smargo开发者联系方式 16](#_Toc499219323)

# 概述

Broker消息中转角色，负责存储消息，转发消息，一般也称为 Server。在 JMS 规范中称为 Provider。Broker通过自身实现方法并且发布，提供Client调用。也就是说相对于Client，Broker是一个Service。

Broker在Smargo的角色很多，它是给Producer、Consumer提供服务的Service、又是调用Registry服务的Client、而它自身还承载着运维接口、消息统计等一系列的功能。

# Borker模块交互

## Registry

◼每个Broker与每个Registry保持长连接。

◼启动时会向每一个Registry注册，启动过后Broker每隔30秒向Register发送心跳，注册和发送心跳都包含了将自身的clusterName，brokerName，topic信息发。如果Broker 2分钟内没有发送心跳数据，则断开连接。

◼ Broker挂掉或者断开，Registry会有自动感应，会更新删除该Broker与Topic的关系。

## Client

◼每个Client通过Registry拿到BrokerList地址，Client与BrokerList保持长连接。

◼Producer向Broker发送消息，Broker负者处理解析消息，然后转发到Stroe进行消息持久化。

◼Consumer从Broker拉取消息进行消费，Broker会维护Consumer与Topic之间订阅关系，并且会维护与Topic消费的Offset。

## Net

◼Broker通过Net创建Service（目前端口为10911），注册并发布服务，供Client调用。

◼Broker通过Net创建Client，调用Registry的方法。

## Store

◼Broker收到消息，经过一些列的验证，解析，重新封装后将消息交给Store做后续的处理。

## Broker

◼Broker主节点之间没有交互，主节点与备节点同步Topic信息，Consumer Offset，延迟队列的Offset，订阅关系等。

# 专业术语

## Topic

Topic是一个消息主题，一个在线Producer实例只能对应一个Topic，一个在线Consumer实例可以对应多个Topic，一条消息必须属于一个Topic。

## ConsumerOffset

ConsumerOffset主要记录了Consumer GroupName与Topic每个Queue的消费进度。

## SubscriptionGroup

SubscriptionGroup用来管理订阅组的订阅信息，包含订阅权限、重试队列，重试次数等。

# Broker实现原理

## Topic 管理

◼ 默认Topic

目前Broker启动时会生成六个默认的Topic，OFFSET\_MOVED\_EVENT、SELF\_TEST\_TOPIC、DEFAULT\_TOPIC、BENCHMARK\_TOPIC、集群名称Topic、Broker名称Topic。其中DEFAULT\_TOPIC最为关键，应为Topic的创建会以DEFAULT\_TOPIC为模板进行创建。目前Smargo中DEFAULT\_TOPIC的读写队列默认为16个；并且是一个可读可写Topic。

◼ 持久化

每个Broker会将其下的每一个Topic进行统一的持久化，这些Topic被全部保存到一个以JSON的形式都保存到一个文件中，Smargo保存Topic文件的路径为/当前用户目录下/store/config/topic.json文件。该文件主要保存了每一个Topic的主要信息如：TopicName（topic名称）、ReadQueueNums（读队列个数）、WriteQueueNums（写队列个数）、Perm（topic权限）、Order（是否为顺序Topic）、topicSysFlag（系统标识）。

文件存储内容如下：

{

"topicConfigTable": {

"topicConfigTable": {

"%RETRY%consumerGroupId-example-200": {

"SEPARATOR": "",

"topicName": "%RETRY%consumerGroupId-example-200",

"readQueueNums": 1,

"writeQueueNums": 1,

"perm": 6,

"topicFilterType": 0,

"topicSysFlag": 0,

"order": false

}

}

},

"dataVersion": {

"timestamp": 1511333414604049700,

"counter": 2023

}

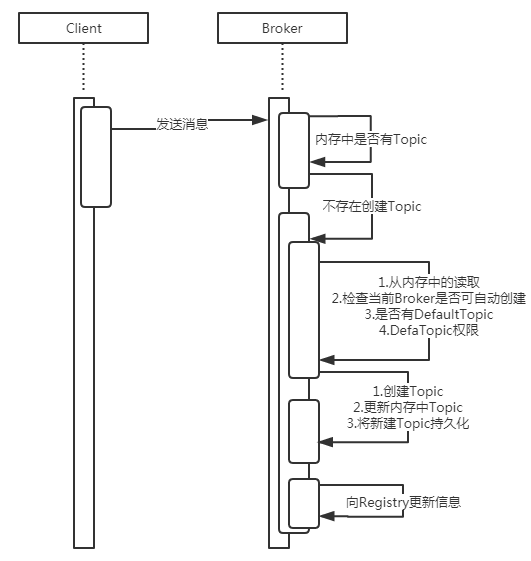
}

◼ 初始化

在Broker启动时，Broker会将Topic.json文件进行加载，在内存中维护一套Topic名称与Topic对象之间的关系，对Topic进行任何操作，都会更新内存所维护的关系以及Topic.json的文件。

◼ 创建Topic

创建Topic由Client发起，Broker没有检测到Client所需要发送的Topic，其创建如图所示：



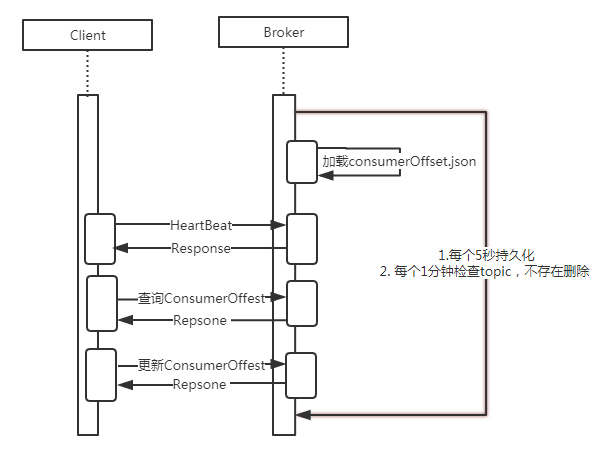
在创建Topic的过程中，会将创建的Topic的队列数与DefaultTopic队列数对比，取其小的队列数为新建Topic的队列数。创建成功后会立马向所有Registry注册。

◼ 其他操作

如果对原有的Topic进行了操作，会第一时间将内存维护的信息进行更新并且会刷入磁盘中。Broker启动时会开启一个线程，每隔30秒向Registry注册，将更新的Topic维护到Registry中。

## ConsumerOffset管理

ConsumerOffset主要管理的是订阅组与Topic Queue消费进度的管理。具体流程如下：



◼ 初始化

在Broker启动时，Broker会将ConsumerOffset.json文件进行加载，在内存中维护一套以订阅组名称与Topic名称组合为key，以当前Topic队列消费offset为value的关系。

◼ 持久化

在Broker启动时候，Broker会开启一个线程每个5秒对Client上报的Consumer与Topic Offset进行持久化。

Smargo保存consumerOffset文件的路径为/当前用户目录下/store/config/consumerOffset.json文件。存储结构如下：

{

"offsets": {

groupName@TopicName: {

queue 1: offset，

…

queue x: offset，

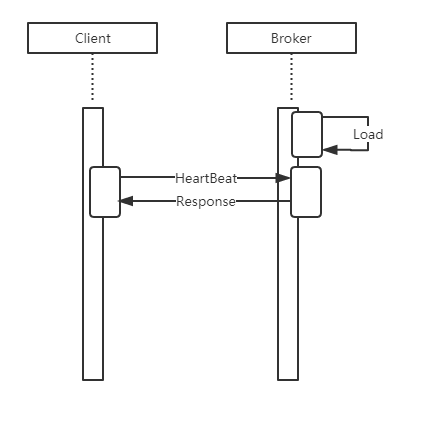
}

}

}

## SubscriptionGroup管理

SubscriptionGroup用来管理订阅组的订阅信息，包含订阅权限、重试队列，重试次数等。其流程如下：



Consumer通过心跳服务进行对SubscriptionGroup来维护。

◼ 持久化

每当Broker维护SubscriptionGroup关系发生改变，都会进行一次持久化。Smargo保存subscriptionGroup文件的路径为/当前用户目录下/store/config/subscriptionGroup.json。存储结构如下：

{

topicName: {

"groupName": "xx", //订阅组名称

"consumeEnable": true, //是否可以消费

"consumeFromMinEnable": true, //是否允许从队列最小位置开始消费，线上默认会设置为false

"consumeBroadcastEnable": true, //是否允许广播方式消费

"retryQueueNums": 1, //消费失败的消息放到一个重试队列，每个订阅组配置几个重试队列

"retryMaxTimes": 16, //重试消费最大次数，超过则投递到死信队列，不再投递，并报警

"brokerId": 0, //从哪个Broker开始消费

"whichBrokerWhenConsumeSlowly": 0 //发现消息堆积后，将Consumer的消费请求重定向到另外一台Slave机器

}

},

"dataVersion": {

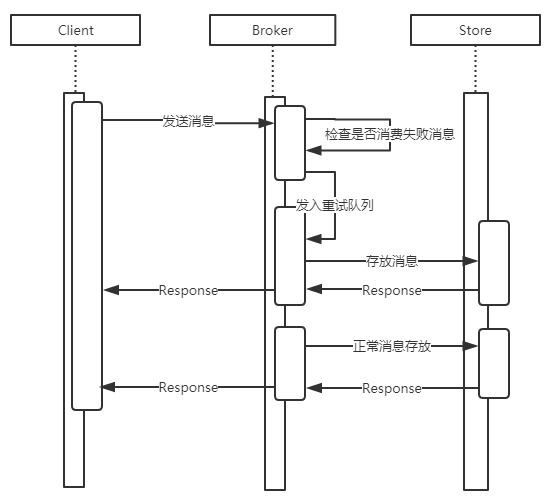
"timestamp": 1511342161274071800,

"counter": 3

}

}

## 发送消息



整个消息的发送流程分为两个步骤流程：

◼ Consumer回退消息

针对Consumer消费失败消息投放重试队列，Broker接收到消息检测到如果是消费失败的Consumer端回退消息。会经历一下流程：

* 1. 检测当前消息的订阅组是否存在。
  2. 检测当前Broker是否有写入权限
  3. 获取到重试队列Topic（重试队列Topic一般为%RETRY%+groupName），计算QueueID。
  4. 如果当前消息消费次数大于设置重试消费次数则投入死信队列(死信队列Topic一般为%DLQ% +GroupName)消息将不会再被消费。
  5. 如果当前消息消费次数小于设置重试消费次数，则会将当前消费次数+3个等级延迟，延迟该消息的消费。
  6. 重新组装消息对象调用store服务。
  7. 统计

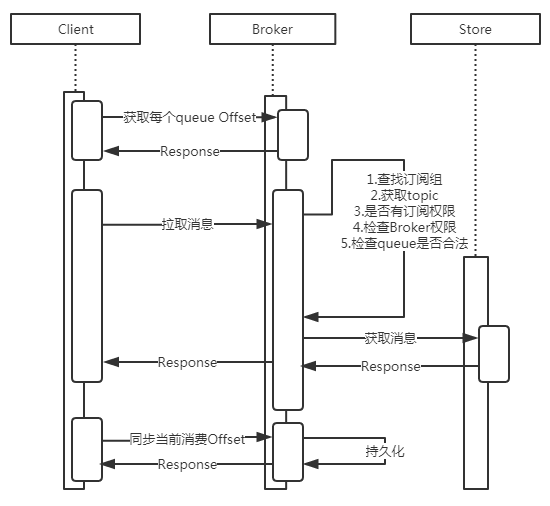
◼ Producter发送正常消息

正常消息的发送情况远没有重试消息流程复杂，其流程如下：

* 1. 检查发送消息的合法性（Topic、Broker权限等）。
  2. 重新组装消息对象调用store服务。
  3. 统计

## 消费消息

Bolt在consumer端的消费方式有两种：一种push（推）、一种pull（拉）。不管是pull与push对Broker而言都是由Consumer主动发起的pull操作。其主要流程如下：



* 1. 在拉取消息时，Consumer会先同步当前Broker持久化当前定于组每个Queue的Offset。
  2. Consumer通过获取到的Offset作为拉取消息的开始位置，向Broker发起拉取消息请求。
  3. Broker校验信息（Broker权限、订阅关系、Topic是否存在、QueueID是否有效）。
  4. 去Store拉取消息。
  5. Broker通过返回结果返回消息。在push模式下如果拉取消息为空，Consumer则会每隔15秒在进行一次拉取。
  6. 进行统计。

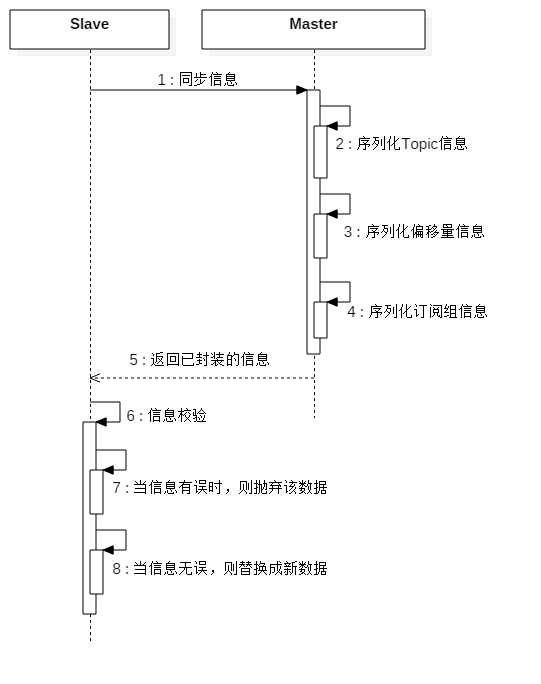
## 主从同步

* 主从关联

Broker集群只允许存在一个主节点，而一个主节点下可挂载一个或多个从节点。主节点与从节点通过心跳保持关联，而从节点彼此间不通信。从节点会定时从Registry查询主节点的地址，当主节点地址发生变化后，原关联就会中断，新的主从关联会重新建立。

* 同步流程

在正常情况下，主节点对外提供Topic创建、消息存储、拉取等服务，而从节点只是用于备份数据。业务Topic从节点定时同步主节点的信息。只有主节点挂了之后，才允许客户端到从节点拉取消息，但是依然不允许从节点存储消息。主从同步时序图如下：



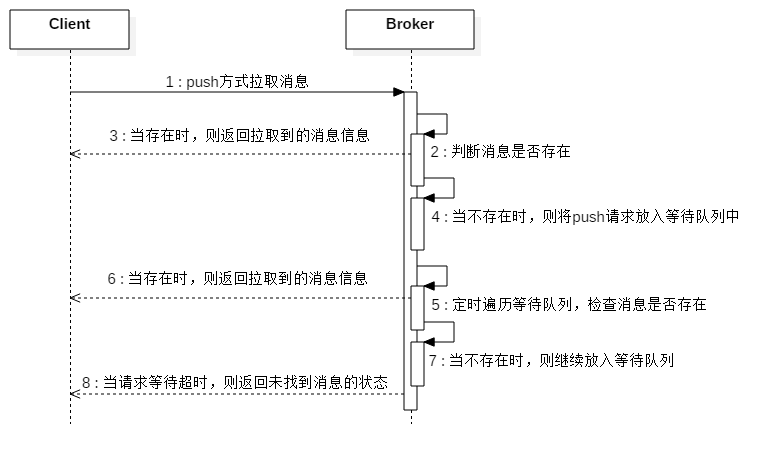
## Hold

* 用途

消费端从Broker拉取消息，当消息不存在时，通过长连接或者定时发送拉取请求来实现当有新消息时能拉取到结果。当新消息到来较迟时，长连接或者定时发送请求的方式，都会造成带宽的浪费，造成Broker的没必要的压力。Broker延缓Hold住拉取消息请求的方式可解决上诉问题，将未拉取到消息的请求放入等待队列，待新消息到来时或者等待超时的唤醒。因Hold服务记录了此次拉取请求的通道信息，所以Broker可从该通道将查询到的消息或者超时状态主动推送给给客户端，从而避免了带宽的浪费。

* Hold流程

消费端从Broker拉取某个偏移量的消息，当该偏移量比最大偏移量都大，则表明该消息还未存入Broker，无法被拉取到。将该拉取请求放入等待队列中，等待有新消息到来时的唤醒。Hold请求的序列图如下：



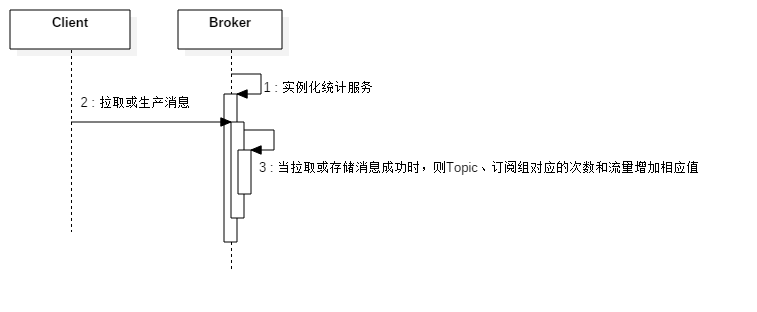
## 消息统计

* 用途

Broker从时间、消息来源、消费等多个维度统计了的Topic、Group的put、get消息的次数、流量等，实时计算、记录了对应的Tps。根据统计数据，用户即可了解自己的业务量、业务波动等，运维即可知道当前Broker集群的处理能力，为扩容等提供可靠数据依据。

* 处理流程

Broker启动时，初始化统计服务，当Broker拉取或存储消息成功时，会在原统计数据的基础上增加相应次数、流量值，计算并记录不同时间段的Tps。消息统计时序图如下：



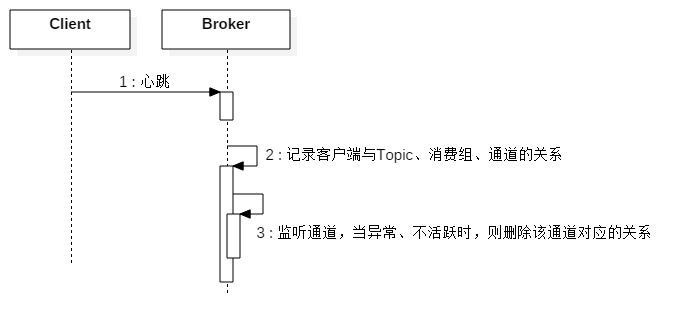
## Producer、Consumer连接管理

* 用途

记录当前生产者、消费者与Broker相连的通道信息，维护通道其与Topic、Group、SubList关系。开发、运维人员可查看当前生产者、消费者数量，Topic对应的客户端数量、同一订阅组下客户端的数量等。

* 连接流程

客户端通过Registry查询到Broker路由信息列表，遍历路由列表，找到业务Topic对应的Broker地址。根据Broker地址，客户端连接Broker，定时发送心跳来维护该通道连接，而Broker接收到心跳后，实时维护Channel、Topic、Group、SubList的彼此之间的关系。客户端连接管理时序图如下：



# 附件一 Smargo开发者联系方式

|  |  |  |
| --- | --- | --- |
| 姓名 | 联系方式 | 更新日期 |
| 谭特贤 | **tantexian@qq.com** | 2017/9/11 |
| 郜焱磊 | **gyl\_adaihao@163.com** | 2017/11/21 |
| 田玉粮 | **idistyl@gmail.com** | 2017/11/21 |
| 尹同强 | [**tongqiangyin@gmail.com**](mailto:tongqiangyin@gmail.com) | 2017/11/21 |
| 罗继 | [**gunsluo@gmail.com**](mailto:gunsluo@gmail.com) | 2017/11/21 |
| 周飞 | **he236555699@163.com** | 2017/11/21 |
| 戎志宏 | **hzrong007@163.com** | 2017/11/21 |