**LavaMQ开发与使用说明书**

文档管理信息表

|  |  |
| --- | --- |
| **主题** | LavaMQ开发与使用说明书 |
| **版本** | 1.0 |
| **内容** |  |
| **关键字** | Lavabox、LavaMQ |
| **参考文档** |  |
| **创建时间** | 2020-01-08 |
| **创建人** | 王雷 |
| **最新发布日期** |  |

文档变更记录表

|  |  |  |
| --- | --- | --- |
| **修改人** | **修改时间** | **修改内容** |
| 王雷 | 2020-01-20 | 初版 |
| 王雷 | 2020-02-18 | 同步更新变更接口相关内容 |
| 王雷 | 2020-04-15 | 更新lava mq服务端与客户端事件介绍 |
| 王雷 | 2020-05-26 | 更新订阅与退订事件 |
|  |  |  |
|  |  |  |

**目 录**

[1 引言 6](#_Toc37837943)

[1.1 编写目的 6](#_Toc37837944)

[1.2 背景 6](#_Toc37837945)

[2 LavaMQ结构设计 6](#_Toc37837946)

[2.1 层级结构 6](#_Toc37837947)

[2.2 结构说明 7](#_Toc37837948)

[2.3 LavaMQ客户端设计 7](#_Toc37837949)

[2.4 LavaMQ服务端设计 7](#_Toc37837950)

[3 接口设计 7](#_Toc37837951)

[3.1 接口设计 7](#_Toc37837952)

[3.2 接口使用说明 8](#_Toc37837953)

[**3.2.1** 配置接口 8](#_Toc37837954)

[3.2.1.1 功能 8](#_Toc37837955)

[3.2.1.2 用法 8](#_Toc37837956)

[3.2.1.3 参数说明 8](#_Toc37837957)

[**3.2.2** 异步请求响应接口 9](#_Toc37837958)

[3.2.2.1 服务端使用接口 9](#_Toc37837959)

[3.2.2.2 客户端使用接口 10](#_Toc37837960)

[**3.2.3** 同步请求响应接口 10](#_Toc37837961)

[3.2.3.1 服务端使用接口 10](#_Toc37837962)

[3.2.3.2 客户端使用接口 10](#_Toc37837963)

[**3.2.4** 订阅推送接口 10](#_Toc37837964)

[3.2.4.1 服务端使用接口 10](#_Toc37837965)

[3.2.4.2 客户端使用接口 11](#_Toc37837966)

[**3.2.5** 快照订阅接口 11](#_Toc37837967)

[3.2.5.1 服务端使用接口 11](#_Toc37837968)

[3.2.5.2 客户端使用接口 11](#_Toc37837969)

[**3.2.6** 事件监听接口 11](#_Toc37837970)

[3.2.6.1 服务端使用接口 11](#_Toc37837971)

[3.2.6.2 客户端使用接口 12](#_Toc37837972)

[**3.2.7** 启停接口 12](#_Toc37837973)

[**3.2.8** 资源释放接口 12](#_Toc37837974)

[4 LavaMQ开发实例 12](#_Toc37837975)

[4.1 环境配置实例 12](#_Toc37837976)

[4.2 异步请求响应实例 13](#_Toc37837977)

[**4.2.1** 服务端开发实例 13](#_Toc37837978)

[**4.2.2** 客户端开发实例 13](#_Toc37837979)

[4.3 同步请求响应实例 14](#_Toc37837980)

[**4.3.1** 服务端开发实例 14](#_Toc37837981)

[**4.3.2** 客户端开发实例 15](#_Toc37837982)

[4.4 订阅推送实例 16](#_Toc37837983)

[**4.4.1** 服务端开发实例 16](#_Toc37837984)

[**4.4.2** 客户端开发实例 16](#_Toc37837985)

[4.5 快照订阅实例 17](#_Toc37837986)

[**4.5.1** 服务端开发实例 17](#_Toc37837987)

[**4.5.2** 客户端开发实例 18](#_Toc37837988)

* **文档撰写及关键评审点**

|  |  |
| --- | --- |
| **对应详设文档章节** | **关键评审点** |
| 0 全局 | **扩展性：**能够考虑到未来业务变化的代码处理设计 |
| 0 全局 | **细节：**对于逻辑复杂的 ，细节描述清晰，需增加对性能的要求 |
| 3 系统结构设计 | **层次：**系统结构层次清晰 |
| 3 系统结构设计 | **耦合：**类关系简单耦合度低 |
| 6 非功能性设计 | **设计：**设计时是否考虑了兼容性、可扩展性、可维护性 |
| 6 非功能性设计 | **接入层：**防攻击设计、伪造、重放、防爬虫、恶意请求等相关设计 |

# 引言

## 编写目的

本文介绍LavaMQ的开发背景，实现原理，以及使用方法和注意事项等，为基于LavaMQ的Java项目组开发人员提供一些参考。

本文的主要读者为各级领导和评审专家，以及基础组件项目组成员、Java项目开发人员、QA测试人员等。

## 背景

目前Java项目组使用的通信模块主要分为三种：Rabbit MQ、MessageBus以及CedaMQ，其中点对点通信使用最多的是CedaMQ，其他两种主要是用于消息总线模式。

考虑到CedaMQ在实际项目使用上可能存在性能与稳定性方面的一些不足，与此同时Lavabox已经可以提供使用，我们根据数据项目组提出的MQ使用需求与推荐的接口，在此基础上经双方讨论后确定了LavaMQ的API接口，并基于Lavabox提供的c++动态库开发了LavaMQ中间件。

LavaMQ提供jar包与接口Api，数据项目组的java开发人员可以据此接入该中间件，并实现同步与异步请求响应、消息订阅、快照订阅等常用的c/s通信功能。

# LavaMQ结构设计

## 层级结构



## 结构说明

LavaMQ开发中的层级结构主要为三层：

1. 底层为lava\_box的C++实现，lava\_box是高性能的通信库，使用它可以快速实现c++版的客户端与服务器之间的点对点通信，通信方式主要支持请求响应(异步或同步)、订阅推送、快照订阅等。
2. 中间层为jni转换层，为实现java端可以顺利使用lava\_box通信库，我们定义好接口文件并使用swig工具自动生成了JNI交互所需的文件，最终将这些文件打为jar包。
3. 上层为LavaMQ，主要是利用JNI接口封装了lava\_box通信库，并实现了一组LavaMQ自定义的接口，数据组的同事可以直接使用LavaMQ提供的这组接口方便的接入，并快速实现java的客户端与服务端通信，通信方式与lava\_box一致。

## LavaMQ客户端设计

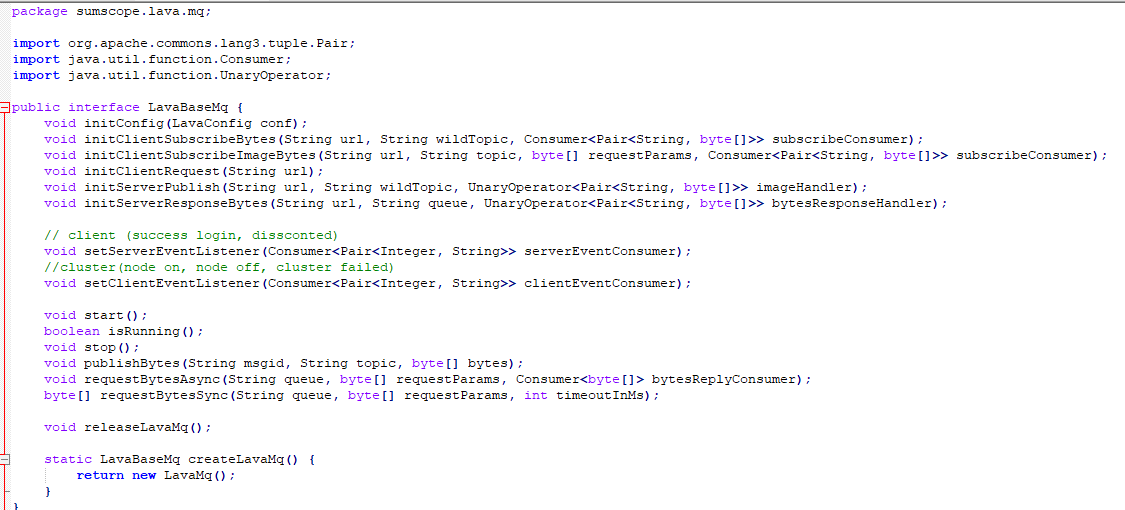
1. LavaMQ的客户端在java层是非共享的，每个LavaMQ的客户端都被创建为一个独立的对象。
2. 每个LavaMQ的客户端内部都会创建一个lava proxy对象，该对象负责与lava\_box交互。
3. LavaMQ内部的lava proxy对象提供的连接方式是共享的，可以使用配置接口设置share session选项来设定是否共享客户端链接。

## LavaMQ服务端设计

1. LavaMQ的服务端是共享的，针对同一个url(注意：url的概念在这里是指服务器自定义的名称，该名称会在zookeeper注册，并非http的URL概念)只会创建一个共享的服务器对象。
2. 每个LavaMQ的服务端内部都会创建一个lava stub对象，该对象负责与lava\_box交互。
3. LavaMQ的服务端提供推送与消息响应功能，针对快照的推送实现了消息顺序机制，保证客户端先收到快照数据，再收到推送的更新数据。
4. LavaMQ服务端多活模式下，可以保证底层多个服务器推送消息的去重，但是并不确定推送消息中会有重复消息，这一点业务层要自己做去重设计。

# 接口设计

## 接口设计



## 接口使用说明

### 配置接口

#### 功能

配置接口为使用者提供设置LavaMQ工作所需的各种参数的功能，使用者要new一个LavaConfig对象，该对象在创建时已经设置好默认参数，使用者如需改变默认参数的值，则直接set该参数即可。

#### 用法

1. 创建一个LavaConfig对象。
2. 设置LavaConfig对象中某些参数的值。
3. 调用initConfig接口将对象引用传入即可完成设置。

#### 参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| 参数类型 | 参数名称 | 默认值 | 功能 |
| boolean | useCluster | true | 是否使用集群，如使用集群则无需设置。 |
| String | host | 127.0.0.1 | 服务端ip，主要是服务端需要设置，客户端如果使用集群则无需设置。 |
| String | port | 10001 | 服务端port，主要是服务端需要设置，客户端如果使用集群则无需设置。 |
| String | nodeInfo | 127.0.0.1:2181 | zookeeper集群ip与port，如果是多个，则使用逗号分隔，服务端与客户端都需要设置。 |
| int | serviceType | 1 | 服务端工作模式  1：多活模式  2：主备模式  其他：不支持 |
| int | asyncThreadCnt | 4 | 异步请求响应线程池中线程数，一般无需更改，如果服务端机器配置很高可以修改的大一点。  建议与CPU核心数匹配 |
| int | postImageThreadCnt | 4 | 快照数据推送线程池中线程数，一般无需更改，如果服务端机器配置很高可以修改的大一点。  建议与CPU核心数匹配 |
| String | userName | 空字符串 | 用户名，用于客户端向服务端认证身份，内部使用一般不需要设置此项。 |
| String | passWord | 空字符串 | 密码，用于客户端向服务端认证身份，内部使用一般不需要设置此项。 |
| boolean | shareSession | true | 是否开启客户端连接线程共享，默认为开启。 |
| boolean | autoConnect | true | 是否开启自动断线重连，默认为开启。  该选项只在非集群模式下有效 |
| long | tryTimes | 3 | 断线重连次数。  该选项只在非集群模式下有效 |
| long | clientIOThreads | 2 | lava\_box环境配置，客户端IO线程数，一般无需更改。 |
| long | serverWorkerThreads | 4 | lava\_box环境配置，服务端工作线程数，一般无需更改。 |
| boolean | useProxy | false | 是否使用代理，默认为不使用。 |
| String | proxyHost | 空字符串 | 代理服务器ip |
| String | proxyPort | 空字符串 | 代理服务器端口 |
| int | proxyType | 1 | 代理类型  1：http  2：socket4  3：socket5  其他：不支持 |
| String | proxyUsername | 空字符串 | 用户名，用于代理认证。 |
| String | proxyPassword | 空字符串 | 密码，用于代理认证。 |

### 异步请求响应接口

#### 服务端使用接口

void initServerResponseBytes(String url, String queue, UnaryOperator<Pair<String, byte[]>> bytesResponseHandler)

功能：初始化服务器工作类型，指定为响应模式，并指定支持响应的queue

参数

url：服务器名称

queue：支持响应的通配或非通配队列名称

bytesResponseHandler：接收客户端请求参数，服务端使用客户端的精确的queue匹配出最佳，然后发送给服务端的业务层去处理，处理后返回byte[]数据给客户端

#### 客户端使用接口

1. void initClientRequest(String url)

功能：初始化客户端类型，指定为请求型客户端

参数

url：服务器名称，指定向这台服务器请求

1. void requestBytesAsync(String queue, byte[] requestParams, Consumer<byte[]> bytesReplyConsumer)

功能：客户端异步请求调用，需要start以后才能使用

参数

queue：请求队列名

requestParams：请求参数，序列化为byte[]

bytesReplyConsumer：客户端消息接收器，接收type[]消息

### 同步请求响应接口

#### 服务端使用接口

同3.2.2.1

#### 客户端使用接口

1.void initClientRequest(String url)

功能：初始化客户端类型，指定为请求型客户端

参数

url：服务器名称，指定向这台服务器请求

2. byte[] requestBytesSync(String queue, byte[] requestParams, int timeoutInMs)

功能：客户端同步请求调用，需要start以后才能使用

参数

queue：请求队列名

requestParams：请求参数，序列化为byte[]

timeoutInMs：超时等待时间，单位为毫秒

返回值

byte[]：返回的响应数据，序列化为byte[]

### 订阅推送接口

#### 服务端使用接口

1. void initServerPublish(String url, String wildTopic, UnaryOperator<Pair<String, byte[]>> imageHandler);

功能：初始化服务端类型，指定为推送模式

参数

url：服务器名称

wildTopic：指定服务器可接收的快照请求的主题，可以是通配的，也可以是非通配的，普通推送指定为null或者””。

imageHandler：快照推送需要业务层传入handler，lava mq服务端回调这个handler并等待业务层返回快照数据，普通推送指定为null。

【注意】：lava mq内部快照数据的推送方式为线程池分拣消息队列，请业务层一定保证handler里面不能永远阻塞，业务层如果处理超时请返回null，如果业务层handler永远阻塞最终会耗尽lava mq线程池中所有线程，并导致队列积压不能正常工作。

1. void publishBytes(String msgid, String topic, byte[] bytes)

功能：服务端推送数据

参数

msgid：多活用于去重的消息id，由服务端的业务层传入

topic：服务端推送的某一精确的topic

bytes：服务端推送的数据，序列化为byte[]

#### 客户端使用接口

void initClientSubscribeBytes(String url, String wildTopic, Consumer<Pair<String, byte[]>> subscribeConsumer)

功能：客户端普通订阅

参数

url：服务器名称，向这台服务器订阅

wildTopic：客户端订阅主题，可以是通配或非通配的

subscribeConsumer：客户端指定订阅消息接收器，Pair中的first为精确topic

### 快照订阅接口

#### 服务端使用接口

同3.2.4.1

#### 客户端使用接口

void (String url, String topic, byte[] requestParams, Consumer<Pair<String, byte[]>> subscribeConsumer)

功能：客户端快照订阅

参数

url：服务器名称，向这台服务器订阅

topic：客户端订阅主题，这里的主题必须是精确的

subscribeConsumer：客户端指定的订阅消息接收器，Pair中的first为精确topic，客户端会先收到一次快照，之后才收到更新数据的推送

### 事件监听接口

#### 服务端使用接口

void setServerEventListener (Consumer<Pair<Integer, String>> serverEventConsumer)

功能：服务端设置事件监听接口，用于接收来自服务器的事件

参数

serverEventConsumer：

Integer为事件定义的编号

String为事件文字描述

注意：服务端设置的listener对象是share的，为同一url的所有server实例所共享，设置一次即可，多次设置的话前面的设置都会被最后一个覆盖

服务端全部事件如下表：

|  |  |  |
| --- | --- | --- |
| Event code | Event message | 事件含义 |
| 13 | cluster register failed! | 服务端向zookeeper注册失败 |
| 14 | cluster node on. | 服务端在zookeeper注册成功，并成为活跃节点，即此服务可以被客户端正常连接 |
| 15 | cluster node off. | 服务端在zookeeper注册成为备用节点，即此服务作为备用服务，在它成为活跃节点前不会被客户端发现并连接 |
| 20 | wildTopic | initClientSubscribeBytes接口传入的wildTopic（通配或非通配）订阅成功后会作为event message返回 |
| 21 | wildTopic | initClientSubscribeBytes类型的mq client对象在stop时会收到退订成功消息 |
| 22 | topic | initClientSubscribeImageBytes接口传入的topic（必须是非通配）订阅成功后会作为event message返回 |
| 23 | topic | initClientSubscribeBytes类型的mq client对象在stop时会收到退订成功消息 |
| 24 | String(具体失败原因) | 普通订阅失败，返回失败原因的message |
| 25 | String(具体失败原因) | 普通退订失败，返回失败原因的message |
| 26 | String(具体失败原因) | 快照订阅失败，返回失败原因的message |
| 27 | String(具体失败原因) | 快照退订失败，返回失败原因的message |

#### 客户端使用接口

void setClientEventListener (Consumer<Pair<Integer, String>> clientEventConsumer)

功能：客户端设置事件监听接口，用于接收来自客户端的事件

参数

clientEventConsumer

Integer为事件定义的编号

String为事件文字描述

注意：客户端设置的listener对象不share，每一个client对象单独维护一个listener对象(如有)

客户端全部事件如下表：

|  |  |  |
| --- | --- | --- |
| Event code | Event message | 事件含义 |
| 8 | login success. | 客户端连接成功 |
| 2 | disconnected! | 客户端连接断开等所有非正常连接状态都返回此事件 |

### 启停接口

1. void start()

功能：用于启动服务端或客户端的工作，一般在调用init函数后调用该函数，否则服务端或客户端不会开始工作。

1. void stop()

功能：用于停止服务端或客户端的工作，一般在已经start的前提下调用，服务端或客户端会停止工作，其他情况下调用无效。

1. boolean isRunning()

功能：用于判断服务端或客户端的工作状态

返回值

true：运行状态

false：停止状态

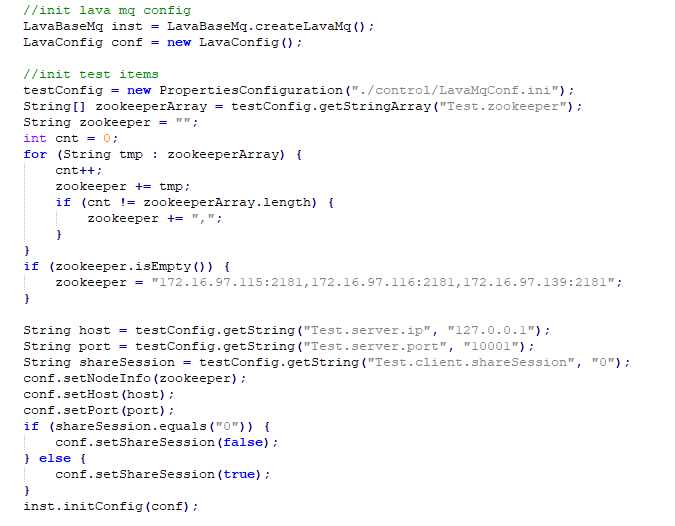
### 资源释放接口

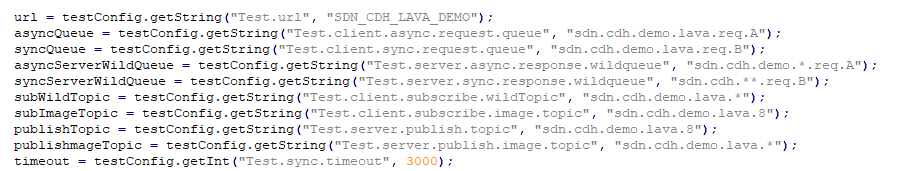
void releaseLavaMq()

功能：释放lava mq相关资源，当完全不再使用lava mq时候调用，该函数会释放lava mq内部一些对象的引用计数、释放lava\_box运行环境等。

# LavaMQ开发实例

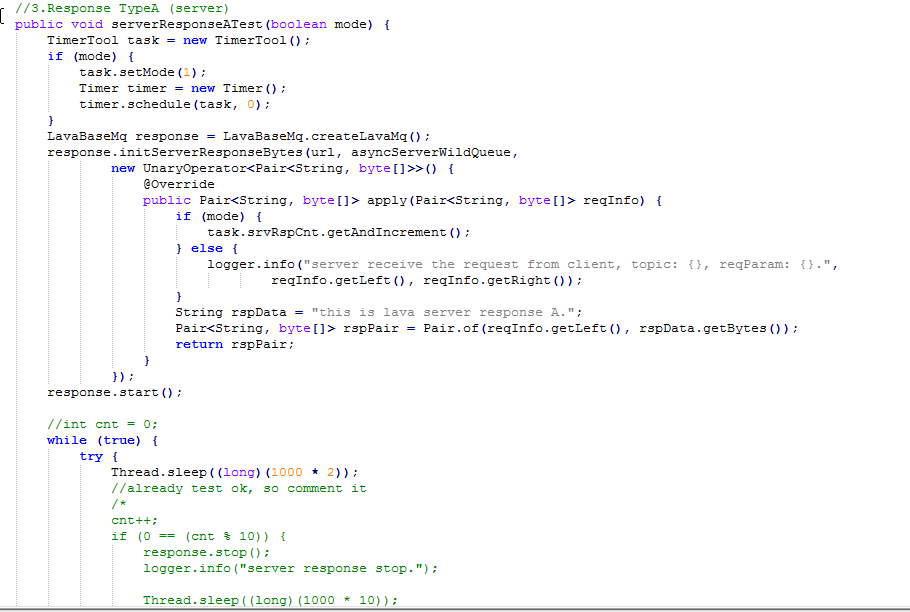
## 环境配置实例



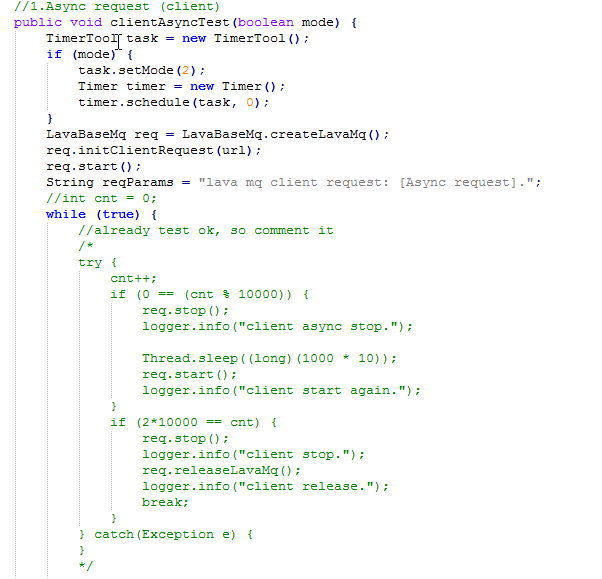


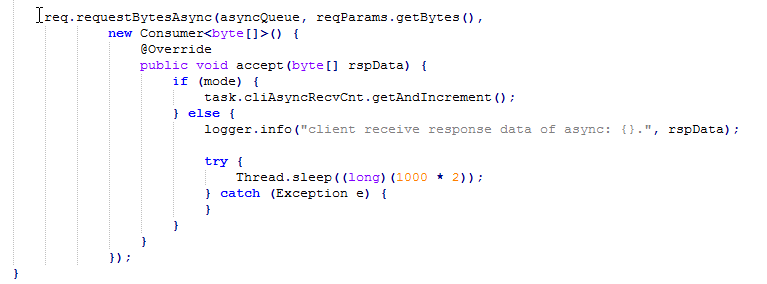
## 异步请求响应实例

### 服务端开发实例



### 客户端开发实例



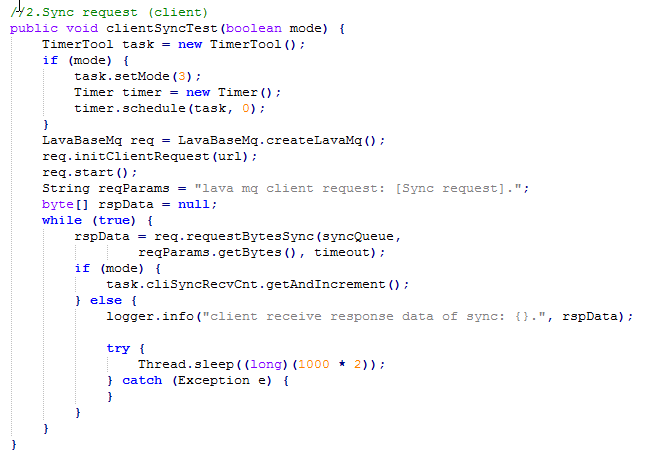


## 同步请求响应实例

### 服务端开发实例

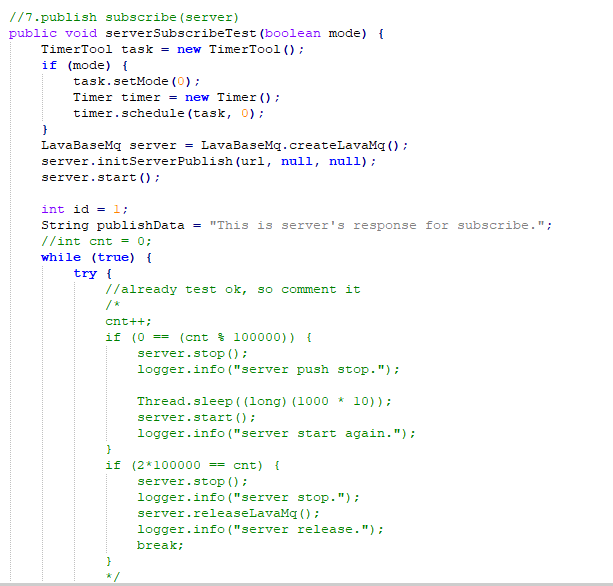


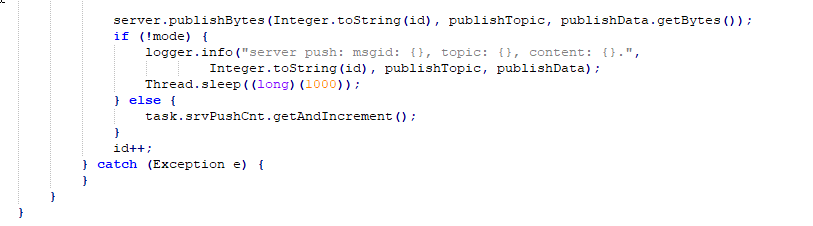
### 客户端开发实例



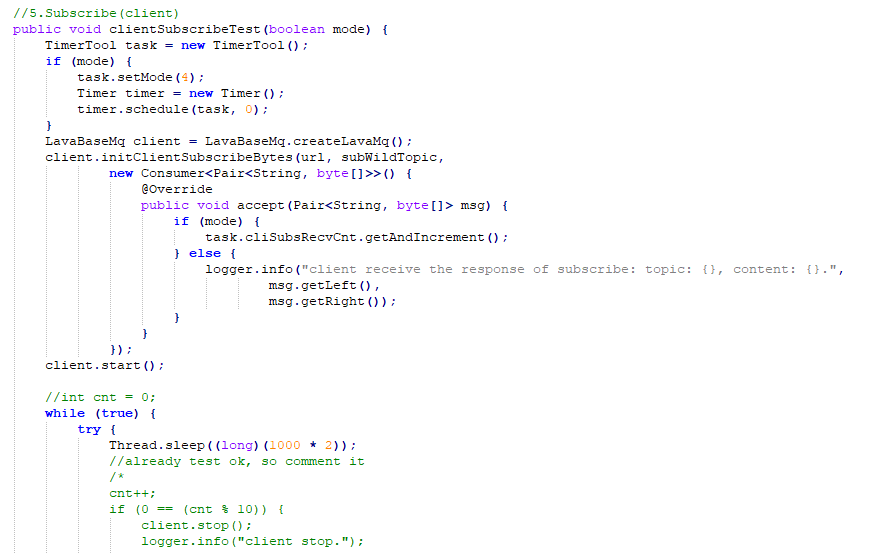
## 订阅推送实例

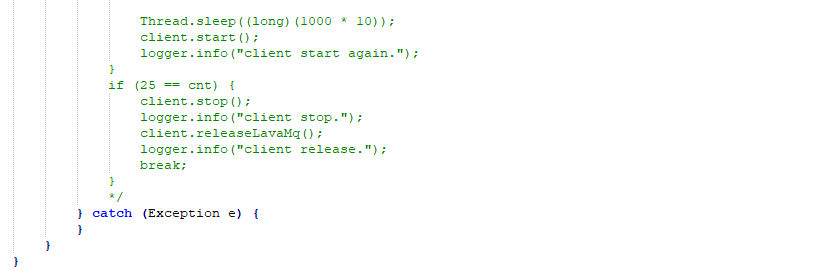
### 服务端开发实例





### 客户端开发实例

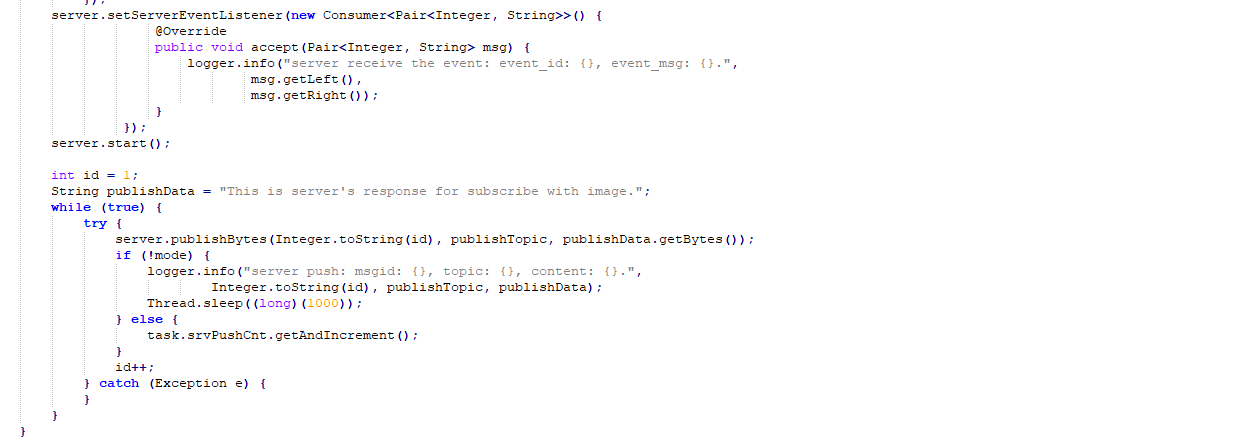




## 快照订阅实例

### 服务端开发实例





### 客户端开发实例

