# KAFKA-656 - Quota Design

## Quota Implementation KAFKA-656

### Requirements:

There are several quantities we would want to track:
1. Requests pers second
2. Bytes written per second
3. Bytes read per second

There are two reasonable groupings we would want to aggregate and enforce these thresholds at:
1. Topic level
2. Client level (e.g. by client id from the request)

When a request hits one of these limits we will simply reject it with a QUOTA_EXCEEDED exception.

### https://issues.apache.org/jira/browse/KAFKA-656

### Conceptual Design:

This is psuedo-code obviously, and incomplete, just starting to put some ideas down.

```
class RequestQuotaclass RequestQuota
{
    public bool checkRequestQuotas(String ClientIP, String Topic, Long BytesToRead, Long BytesToWrite)
    {
        long topicRPSLImit = QuotaConfiguration.getTopicRPSLimit(Topic);
        if (! Quota.checkRPS(Topic, topicRPSLImit))
        {
            return false;
        }

        long clientRPSLImit = QuotaConfiguration.getclientRPSLimit(ClientIP);
        if (! Quota.checkRPS(ClientIP, clientRPSLImit))
        {
            return false;
        }

    }
}
class Quota
{
    public bool checkRPS(String targetEntity, long Limit)
    {
        Meter rps = Metrics.newMeter(this.class, 'rps-' + targetEntity, 'requests', TimeUnit.SECONDS);
        rps.mark();

        if (Limit > 0 && rps.getOneMinuteRate() >= Limit)
        {
            return false;
        }
        return true;
    }
}
```