# ServerId设计篇

## serverId介绍：

serverId在整个服务中扮演了非常重要的作用，数据的存储和读取，以及副本恢复都和serverId有着直接的关联。

**ServerId的作用：**

1. 存储时，标识某个server所管理的文件；
2. 读取时，快速定位相应的server，来查询文件；
3. 副本恢复时，记录文件迁移规则。

服务端在启动的过程中，会自动生成serverId。每个服务端会生成两种serverId，一个是对应单个副本的serverId，一个是对应多个副本的serverId。

1. 对应单个副本的serverId，在生成后是一直不会变的，因为副本数为1，不存在副本迁移的情况。
2. 对应多个副本的serverId，在生成后会有过期时间，如果某个server挂了，并且副本恢复完成，则该serverId会过期，并会生成新的serverId，之前的serverId所管理的数据可以通过迁移规则来查找。

## Serverid生成：

serverId是需要保存到fid（详情见fid设计篇）中，编码后返回给客户的。由客户以fid来查找数据。

之前想用guid来生成serverId，不过guid的长度有点长。因此想尝试用其他的方式来生成serverId。

建议：因serverId只是在启动时会检验或生成，所以可以采取zk的一个序列节点来专门维护serverId的生成策略。

## Serverid保存：

在服务所在的根目录的meta/目录里，会有single.meta文件和multi.meta文件。里面分别存放对应1个副本的serverId和对应多个副本的serverId。

服务端在启动的时候，会判断single.meta文件和multi.meta文件是否存在，以及是否可以读取到相应的serverId。

对于单个副本，

若存在serverId，则使用该serverId，

若不存在，则新生成。

对于多个副本，

若存在serverId，则检查serverId是否过期，

若无过期，则使用该serverId，

若过期，则重新生成serverId。

若不存在serverId，则生成新的serverId。

服务启动成功后，会将serverId等信息，记录到zk中自己创建的临时节点中。

## serverId主要接口设计：

判断一个多副本serverid是否过期

Boolean expireFlag=MetaManager.isMultiServerIdExpire()

返回值：true为serverid过期，false为不过期。

注意：因副本恢复后，会建立副本恢复规则，若副本恢复规则已建立，则可以视为多副本serverid已过期。

生成一个serverid

String serverId=MetaManager.genServerId(Enum serverIdType);

参数：serverIdType为枚举类型，值可为singleServerId和multiServerId。

返回值：为生成的serverId。

注意：此处可以在zk上生成一个维护自增serverid的节点，如果有server进行启动，则需要借助分布式锁来生成serverId。