Agent或最终目的地的数据，这也是避免发送数据到下一阶段的调节机制。在Flume框架中，为了正确地运行，process方法必须是线程安全的。例5-12给出一个自定义Sink的例子。

例5-12 一个自定义Sink的例子

```
package usingflume.ch05;

public class S3Sink extends AbstractSink implements Configurable {
  private String objPrefix;
  private final AtomicLong suffix = new AtomicLong(System
    .currentTimeMillis());
  private String awsAccessKeyId;
  private String awsSecretKey;
  private String bucket;
  private int batchSize;
  private String endPoint;
  private int bufferSize;
  private AmazonS3 connection;

  // 64K buffer
  public static final int DEFAULT_BUFFER_SIZE = 64 * 1024;
  public static final int DEFAULT_BATCH_SIZE = 1000;
  public static final String DEFAULT_OBJECT_PREFIX = "flumeData-";

  @Override
  public void start() {
    // Set up Amazon S3 client
    AWSCredentials credentials = new BasicAWSCredentials(
      awsAccessKeyId, awsSecretKey);
    ClientConfiguration config = new ClientConfiguration();
    config.setProtocol(Protocol.HTTP);
    connection = new AmazonS3Client(credentials, config);
    connection.setEndpoint(endPoint);
    if (!connection.doesBucketExist(bucket)) {
      connection.createBucket(bucket);
    }
    super.start();
  }

  @Override
  public synchronized void stop() {
    super.stop();
  }

  @Override
```