> 败那么没有办法知道数据是否是有效的，因为可能在一行中的最后一列的中间发生写失败。这样的失败会导致异常数据被处理。另一方面，二进制格式将无法读取一个不完整的写入，通常会用一个异常来允许读取数据的应用程序指导数据是异常的且可以跳过。在这种情况下，读取数据的应用程序可以直接移动到下一个文件。为了被更好地检测到错误，二进制格式通常被用于从 Flume 中写出数据。
>
> 如 RCFile、ORCFile、Parquet 等二进制柱状格式怎么样？大多数列式格式对于批量写是优化了的，即在一个写操作中写大批量的数据。另一方面，Flume 按照它传入的数据写数据。可能对于这样的格式不是非常适合。在 Flume 中使用像 Parquet 的格式的一个好方法是，以 Avro 格式写数据，然后使用 Parquet 自带的工具或使用 Impala 把它转换成 Parquet。

HDFS 序列化器通常是一个很简单的类，使用 Flume 配置系统进行配置。序列化器类本身必须实现 `org.apache.flume.serialization.EventSerializer` 接口。Flume 需要使用继承自 `org.apache.flume.serialization.EventSerializer.Builder` 的 builder 类创建该类。所有的 builder 类必须有一个无参的公共构造方法，当实例化 builder 类时，Sink 使用该构造方法。

例 5-1 中展示了 `EventSerializer` 接口。

例5-1 EventSerializer接口

```
package org.apache.flume.serialization;
public interface EventSerializer {
  public static String CTX_PREFIX = "serializer.";
  public void afterCreate() throws IOException;
  public void afterReopen() throws IOException;
  public void write(Event event) throws IOException;
  public void flush() throws IOException;
  public void beforeClose() throws IOException;
  public boolean supportsReopen();
  public interface Builder {
    public EventSerializer build(Context context, OutputStream out);
  }
}
```

HDFS Sink 传递一个 `OutputStream` 实例和 `Context` 实例给 builder，会依次创建和配置返回给 Sink 的 `Serializer` 实例。序列化器被期待转换传递到它 `write` 方法的事件为需要的格式，然后将数据写出到提供的输出流中。序列化器可以使用传递的 `Context` 实例进行配置。

110