中读取的数据。如果写入 Channel 失败，并且事件不能被写入，流 Directory Source 就能重读事件。反序列化器可以使用 mark 和 reset 方法的这个功能，以确保它在能回滚到 stream 正确的位置。 54

反序列化器实现另外两个方法，即 Source 从流读取事件要调用的方法——readEvent 和 readEvents 方法。readEvent 方法必须从流中返回一个事件，而 readEvents 方法接受一个参数，这个参数是从流中读取事件的最大值。

例 3-9 展示了一个反序列化器，基于例 3-8 所示的格式，反序列化消息序列化为 Protocol Buffer (Protobuf) 消息。每个 Protobuf 先将它的 4 字节整数的长度写入文件，再将消息写入文件。

例3-8 ProtobufDeserializer使用的Protobuf格式

```
option java_package = "usingflume.ch03";
option java_outer_classname = "UsingFlumeEvent";

message Event {
  repeated Header header = 1;
  required bytes body = 2;
}

message Header {
  required string key = 1;
  required string val = 2;
}
```

例3-9 ProtobufDeserializer:反序列化写入的数据为Protobuf消息

```
package usingflume.ch03;

public class ProtobufDeserializer implements EventDeserializer {
  private final ResettableInputStream stream;
  private boolean isOpen;

  private ProtobufDeserializer(ResettableInputStream stream) {
    // No configuration to do, so ignore the context.
    this.stream = stream;
    isOpen = true;
  }

  @Override
  public Event readEvent() throws IOException {
    throwIfClosed();
    // To not create an array each time or copy arrays multiple times,
```