```
    // read the data to an array that backs byte buffers,
55  // then wrap that array in a stream and pass it to the Protobuf
    // parseDelimitedFrom method.
    // The format is expected to be:
    // <length of message> - int
    // <protobuf message (written using writeTo (not delimited)>
    // We assume here that the file is well-formed and the length
    // or the
    // message are not partially cut off.
    byte[] sz = new byte[4];
    if (stream.read(sz, 0, 4) != -1) {
      int length = ByteBuffer.wrap(sz).getInt();
      byte[] data = new byte[length];
      stream.read(data, 0, data.length);
      UsingFlumeEvent.Event protoEvent =
        UsingFlumeEvent.Event.parseFrom(new ByteArrayInputStream(data));
      List<UsingFlumeEvent.Header> headerList
        = protoEvent.getHeaderList();
      Map<String, String> headers = new HashMap<String, String>(
        headerList.size());
      for (UsingFlumeEvent.Header hdr : headerList) {
        headers.put(hdr.getKey(), hdr.getKey());
      }
      return EventBuilder.withBody(protoEvent.getBody().toByteArray(), headers);
    }
    return null;
  }

  @Override
  public List<Event> readEvents(int count) throws IOException {
    throwIfClosed();
    List<Event> events = new ArrayList<Event>(count);
    for (int i = 0; i < count; i++) {
      Event e = readEvent();
      if (e == null) {
        break;
      }
      events.add(e);
    }
    return events;
  }

  @Override
  public void mark() throws IOException {
    throwIfClosed();
```