户转换 Flume 事件为目的地系统所需的格式：假设为 HBase *Put* 和 *Increment*。这可以借助序列化器来实现。每个 Sink 有自己的序列化器接口；它们彼此之间稍有不同，主要是使用的表示写入 HBase 的 API 略有不同。我们先讨论嵌入到 Async HBase Sink 的序列化器接口。例 5-4 展示了 Async HBase Sink 接口，每个方法后有解释。HBase Sink 的序列化器与此很相似。

例5-4 Async HBase Sink序列化器接口

```
package org.apache.flume.sink.hbase;
public interface AsyncHbaseEventSerializer extends Configurable,
ConfigurableComponent {
  public void initialize(byte[] table, byte[] cf);
  public void setEvent(Event event);
  public List<PutRequest> getActions();
  public List<AtomicIncrementRequest> getIncrements();
  public void cleanUp();
}
```

尽管在接口中不明显，`AsyncHbaseEventSerializer`（和 `HbaseEventSerializer`）有能力从 Flume 配置系统接收配置。任何通过配置文件传递的配置参数都会传给序列化器的 `configure` 方法（继承自 `Configurable` 接口）。当 Sink 启动时，Sink 会创建序列化器的实例然后调用 `initialize` 方法，传递给它配置中设置的 `table` 和 `columnFamily` 值。

一旦 Sink 从 Channel 中读完一个事件，就调用序列化器的 `setEvent` 方法并且传给事件。之后，Sink 调用 `getActions` 方法，会返回 `PutRequest` 对象列表 [put-request]（Async HBase 相当于 HBase Put），然后调用 `getIncrements`，会返回 `AtomicIncrementRequest` 对象列表 [increment-request]（Async HBase 相当于 HBase Increment）。

很明显，每个 Flume 事件可以产生零或多个 HBase Put 和零或多个 HBase Increment。这允许用户非常灵活地解析事件，数据是基于写入多个行或列和多个计数器递增的。例 5-5 所示的 `AsyncHBaseDirectSerializer` 确实是做到了。这个序列化器查找三个报头：`rowKey`、`incrementColumns` 和 `payloadColumn`。如果 `rowKey` 报头不存在，那么就忽略事件。如果存在 `incrementColumn` 报头，它被看作为一个以逗号分隔的字符串列表，按照 `rowKey` 报头指定的行递增这些列。如果存在 `payloadColumn` 报头，事件主体写入到 `rowKey` 报头指定的行的列中。这个序列化器不接受任何配置，但是可以通过配置文件传递这些配置给序列化器。配置传递给序列化器的 `configure` 方法。

118

例5-5 Async HBase Sink序列化器的例子

```
package usingflume.ch05;

public class AsyncHBaseDirectSerializer
```