Event-driven Source 比 Pollable Source 稍微更复杂一些，因为 Source 必须追踪产生数据的外部程序，并且不借助 Flume 框架处理传入的数据。如果 Event-driven Source 实现了 Configurable 接口，也可以通过配置文件进行配置。

要写 Event-driven Source，需要继承 AbstractEventDrivenSource 类，其已经是实现了 Configurable 接口。或者，可以只是简单地实现标记接口 EventDrivenSource，其继承了 Source 接口。因此，Source 需要实现的唯一的方法就是 Source 接口定义的方法，如果需要，还要实现 Configurable 接口的方法。

AbstractEventDrivenSource 继承了 BasicSourceSemantics 类，其实现了 start、stop 和 configure 方法。这些调用获得了这个类的 doStart、doStop 和 doConfigure 方法的授权。

框架一旦启动了 Source，Source 就自行工作，直到被框架停止。这完全取决于 Source 的实现者，它创建线程处理生成 Flume 事件的外部事件。

例 3-15 展示了 Event-driven Source 经过 Netty-Avro IPC 接收外部服务数据的一个例子。我们在这里不会详细讨论 IPC 协议，但基本思想是，处理程序类，在这种情况下 TransactionSource 类必须实现一个 FlumeCreditCardAuth 接口，其由协议定义文件中的 Avro 生成，如下所示：

75

```
@namespace("usingflume.ch03")

protocol FlumeCreditCardAuth {

  record CreditCardTransaction {
    string cardNumber;
    string location;
    float amount;
  }

  enum Status {
    OK,
    FAILED
  }

  Status transactionsCompleted(array<CreditCardTransaction> transactions);
}
```

Avro 编译器生成代表 CreditCardTransaction 的类和表示回调的接口，当客户端发送数据给这台主机时，调用回调的 transactionCompleted 方法。TransactionSource 实现了这个接口。关于 Avro IDL 的信息可以在 Avro 文档 [avro-idl] 中查找。