```
    throws AvroRemoteException {
    Status status;
    List<Event> events = new ArrayList<Event>(transactions.size());
    for (CreditCardTransaction txn : transactions) {
      StringBuilder builder = new StringBuilder();
      builder.append(txn.getCardNumber()).append(DELIMITER)
        .append(txn.getLocation()).append(DELIMITER)
        .append(txn.getAmount()).append(DELIMITER);
      events.add(EventBuilder.withBody(builder.toString().getBytes(
        Charsets.UTF_8)));
    }
    try {
      getChannelProcessor().processEventBatch(events);
      status = Status.OK;
    } catch (Exception e) {
      status = Status.FAILED;
    }
    return status;
  }
}
```

Source 需要一些配置参数，如主机名和服务器必须绑定的端口。当 Source 初始化且这个 Source 所有的指定配置都通过 Context 实例传递完，`configure` 方法被 Flume 框架调用。在这个方法中，所有相关配置必须被验证并被保存到相关字段。例如，TransactionSource 的 doConfigure 方法验证配置文件中用户提供的主机名和端口，如果它们没有指定则抛出异常。

配置一旦完成，框架通过调用 `start` 方法启动 Source，授权该类的 doStart 方法。在这个方法中，Source 启动服务器进程。一旦开始，框架不与 Source 进行交互直到它被停止。任何服务器或将要接收数据的线程都在这个方法中被启动。对于 `TransactionSource`，在这个方法中启动 NettyServer。

NettyServer 管理多个通过网络接收数据的线程。当接收到一个完整 Avro 消息时，它调用 `transactionCompleted` 方法，来传递数据。该方法将 Avro 格式的数据转换为 Flume 事件。作为演示代码，翻译只是将字符串格式版本的数据表示为字节。这种转换可以包括任意逻辑，尽管保持逻辑简单是一个好主意，因为代码将执行收到的每个事件，并直接影响 Source 的性能。

传入的数据一旦被转换为 Flume 事件，整个 Flume 的批量事件通过 processEventBatch 方法传递到 Channel 处理器。Channel 处理器处理关于 Channel 的事务，如果提交到任何 Channel 失败了就抛出一个异常，在这种情况下，Source 会给发送者返回 `Status.FAILED`，发送者可能会再次发送相同的数据，以确保数据被持久化到 Flume Channel。