Logger Sink 记录到 *log4j.properties* 文件中配置的 Flume Agent 的 log4j 日志文件。该类 Sink 所有的配置都从 *log4j.properties* 文件中获取，且它不需要任何其他的参数（除了 type 和 channel 参数）。表 5-13 展示了 Logger Sink 的配置参数。

表5-13 Logger Sink配置

135

参数	默认值	描述
type	-	Logger Sink 的别名是 logger，也可以使用 FQCN org.apache.flume.sink.LoggerSink

编写自定义 Sink*

在许多情况下，用户将很可能需要编写自定义的 Sink。这种情况下的一个例子是，如果用户需要将数据写入专有数据存储或自定义格式。在本节中，我们将介绍 Sink 的基本工作流，以及编写一个自定义 Sink 的例子。

自定义 Sink 必须实现 Sink 接口，如果 Sink 需要接受来自配置系统的配置，那么可以选择实现 Configurable 接口。为了更好地理解如何编写一个 Sink，很重要的是要理解 Flume 框架如何与 Sink 交互。

当 Agent 启动时，框架检查来确保每个 Sink 有一个指定的 type，且有一个 Channel 参数的值代表了 Agent 中已经存在的配置合适的 Channel。然后 Sink 被实例化且配置传递给它的 configure 方法。如果 configure 方法失败且抛出异常，那么该 Sink 会从 Agent 中移除且删除它的实例。一旦 Sink 成功配置，它就连接到应该从其中读取事件的 Channel。

从那时起，Sink 由一个 Sink 运行器管理。Sink 运行器只是一个负责运行该 Sink 的线程。框架通过调用 start 方法框架来启动 Sink。如果 start 方法失败，该框架将反复重试启动 Sink。

一旦 Sink 启动，Sink 运行器线程就循环调用 process 方法。该方法负责从 Channel 读取数据并写出到下一阶段或最终目的地。每个 process 调用必须处理整个事务——启动事务，从 Channel 读取事件，提交或回滚事务，并最终关闭事务。如果 Channel 不包含任何 Sink 可以移除的事件，process 方法必须返回 Status.BACKOFF，这使得 Sink 运行器只在一个时间间隔后重试，增加每个连续时间 Sink 返回 Status.BACKOFF 时。这种机制减缓了 Sink 在没有足够的数据。如果 Sink 成功，它必须返回 Status.READY 状态，且运行器将立即再次调用 process 方法。

当从 Channel 读取数据或写入到目的地时，如果遇到一些异常，Sink 必须返回 Status.BACKOFF 状态或抛出异常以报告失败。这使 Sink 运行器减慢速度，如果不能清除下流

136