Context 实例接收配置传递给 builder，进而可以通过构造函数传递到反序列化器实例。例 3-7 展示了 Spooling Directory Source 使用 ProtobufDeserializer 进行配置。

Flume 自带了一部分反序列化器。默认的反序列化器是 LineDeserializer[line-deserializer]。这是反序列化器的一个示例配置。如果 Spooling Directory Source 没有设置反序列化器或 deserializer 参数的值设置为 line，行反序列化器将启用。行反序列化器逐行读取文件，基于一个可配置的字符集（默认 UTF - 8）将每一行转换为一个事件。表 3-7 列出了配置参数。

57

表3-7 行反序列化器配置

参数	默认值	描述
outputCharset	UTF-8	转换从文件中读取的字符为字节数组要使用的字符集，这是为事件主体设置的
maxLineLength	2048	每行返回的最大字符数。如果行比这个参数大，则截断来返回

另一个与 Flume 自带的反序列化器是 AvroEventDeserializer。要使用这个反序列化器，设置 deserializer 参数的值为 avro。Avro 反序列化器可以读取 Avro 容器文件并且发送 Avro 格式事件的数据。这个反序列化器只有一个配置参数，如表 3-8 中描述。

表3-8 Avro反序列化器配置

参数	默认值	描述
schemaType	flume.avro.schema.hash	该项参数可以设置为 flume.avro.schema.hash 或 flume.avro.schema.literal。设置为 flume.avro.schema.hash 将导致模式的 64 位 Rabin 指纹使用 flume.avro.schema.hash 的关键词插入 header。如果该参数的值设置为 flume.avro.schema.literal，整个 JSON 化模式将使用 flume.avro.schema.literal 关键词插入到 header

解释数据，重要的是要知道所使用的模式。虽然这些信息包含在文件本身中，保持每个事件的模式也是很重要的，这样可以从事件中读取数据。反序列化器支持在 header 插入 Avro 模式，或简单地将该模式的 64 位 Rabin 指纹（正如 Avro 规范 [schema-fp] 中所指定的）插入 header，这在后面可以用来查找被模式指纹索引的模式注册表（Avro 格式的数据将存入几个已知的模式之一，这是最常见的场景。在 header 中使用指纹允许用户指明读取信息时应该使用的模式，这样当写数据的时候就能被反序列化器所使用）。要想在事件的 header 中设置指纹，则设置 schemaType 参数的值为 flume.avro.schema.hash。