Question 2

Our approach to building the dataflow in Apache Nifi was to build the pipeline starting from the source and gradually add on and test processors until complete.

Tailfile was the first processor used. It streams all new lines that are added to a specific file into the pipeline for processing.

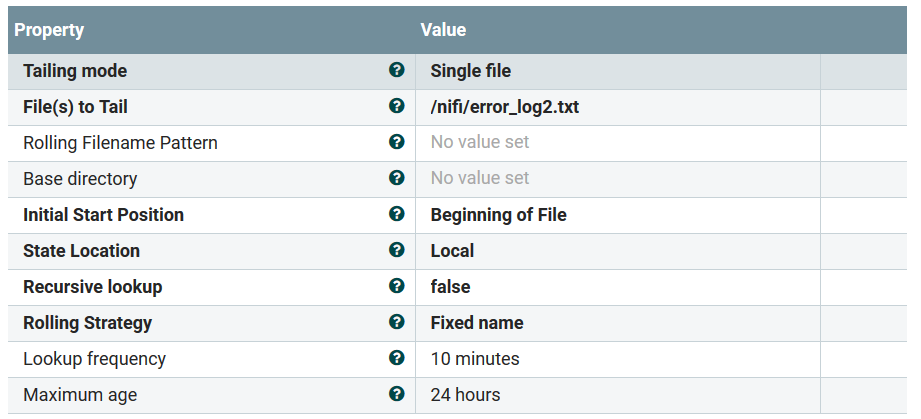


Figure : Tailfile processor properties

SplitText is the second processor used. It takes all lines passed by the TailFile processor and ensures that only a single line is contained in each object in the pipeline. Only successfully split lines are passed to the next pipeline stage, failures and original lines are automatically terminated.

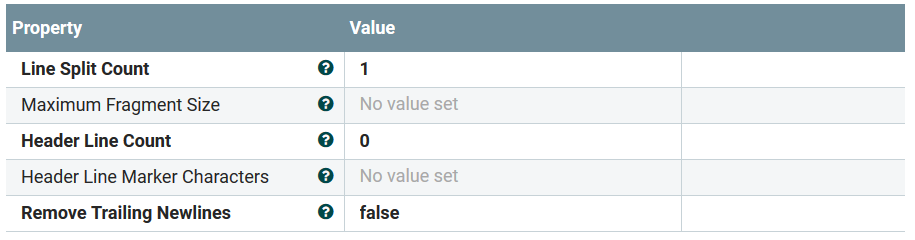


Figure : SplitText processor properties

RouteOnContent is the next processor used. For each line that is passed, this processor checks if the line contains certain keywords, and routes them to a subsequent processor based on the matched keywords. If the line contains “error”, “Error” or “fail”, it will be routed to path 1. If the line contains “Warn” or “Warning”, it will be routed to path 2. If the line contains none of these keywords, it will be discarded.

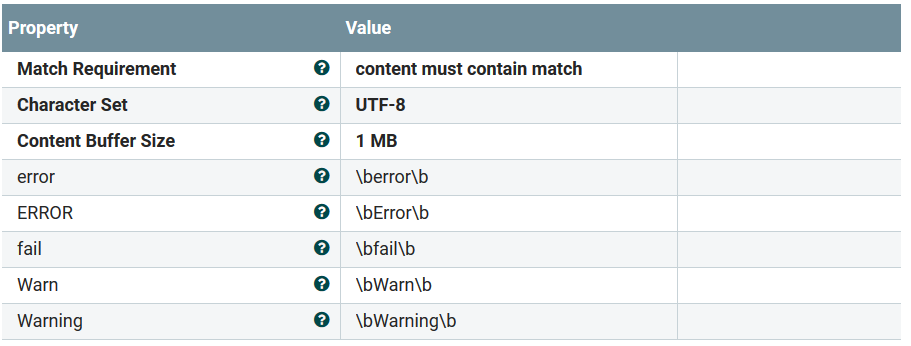


Figure : RouteOnContent processor properties

PublishKafka\_0\_10 is the next processor used. It is used as a Kafka producer, meaning that it takes an input stream at places it into a Kafka topic for later consumption by a Kafka consumer. Lines containing “error”, “Error” or “fail” will be routed to the “Error’” topic on Kafka. Lines containing “warn” or “Warning” will be routed to the “Warn” topic on Kafka.

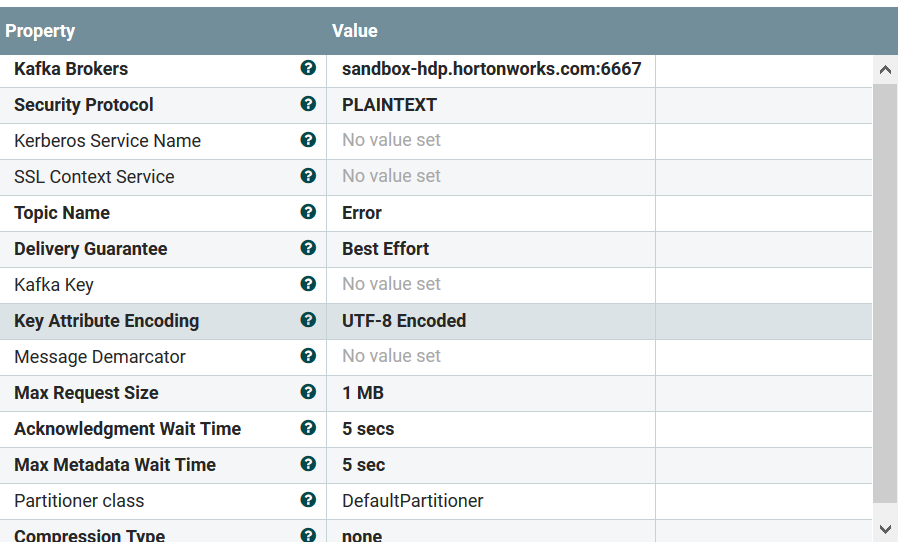


Figure : PublishKafka\_0\_10 properties

ConsumerKafka\_0\_10 is the next processor used. It is used to consume records from a Kafka topic. It’s first used to stream from the “Warn” and “Error” Kafka topics before Flume sinking to HDFS. It is also used to stream from the “Results” topic before sinking those records to Cassandra and HDFS.

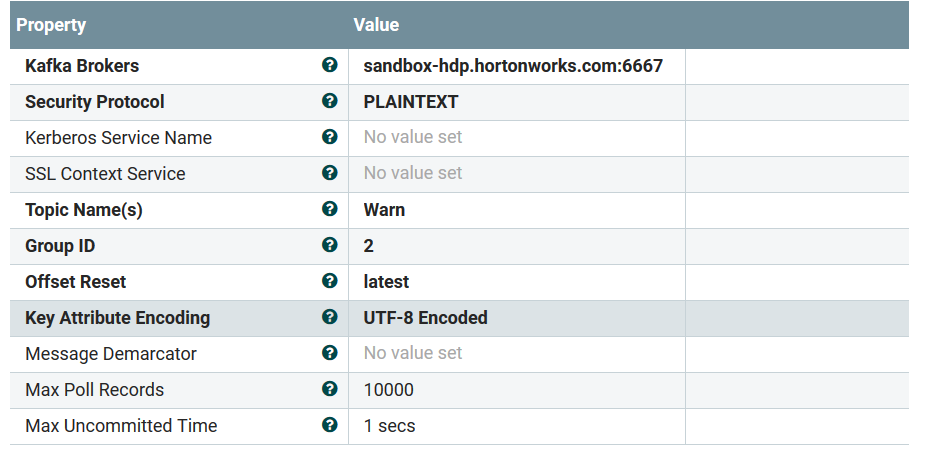


Figure : ConsumerKafka\_0\_10 processor properties

ExecuteFlumeSink is used to sink log streams into HDFS. It is used for the Kafka “Warn” topic and “Error” topic to stream all records into one specific file for each topic.

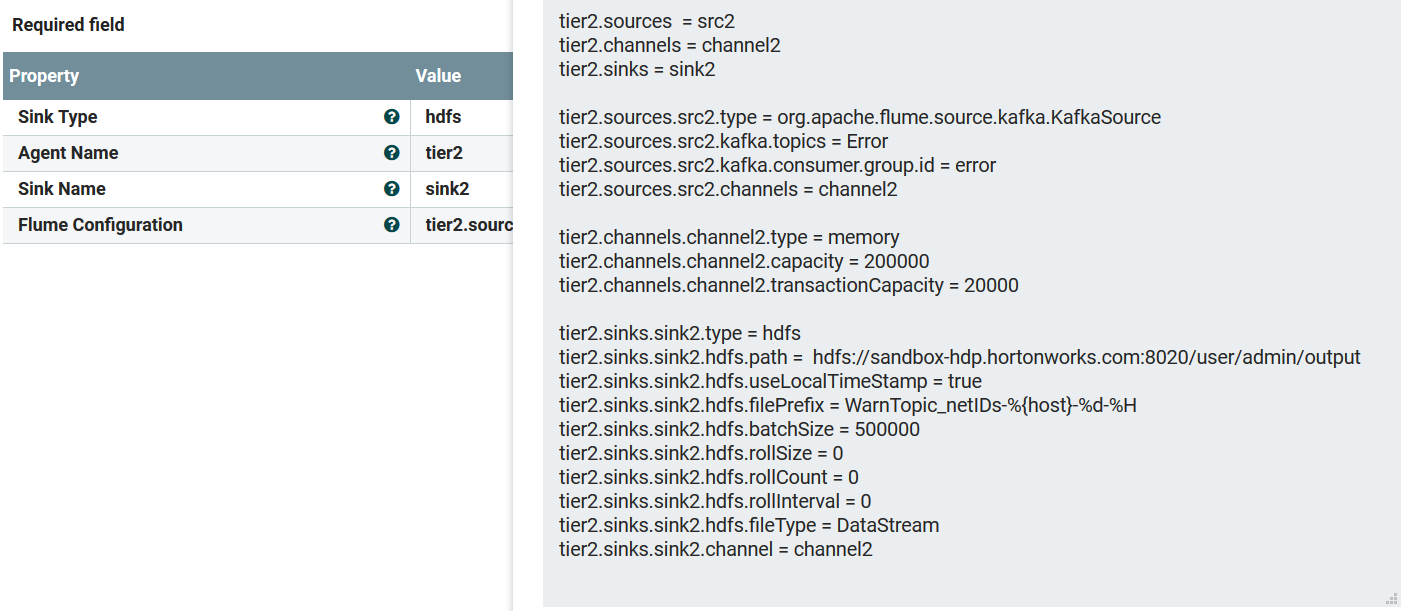


Figure : ExecuteFlumeSink processor properties

Before sinking from the Kafka “Results” topic into HDFS, the filename attribute has to be set for each record passing through the pipeline. This is accomplished using the UpdateAttribute processor.

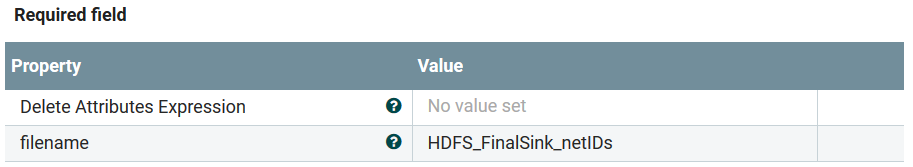


Figure : UpdateAttribute processor properties

PutHDFS is used to place the now renamed records from Kafka “Results” into HDFS. It is set to append all records to a file of the same name. It is an endpoint of the pipeline and discards all records after insertion.

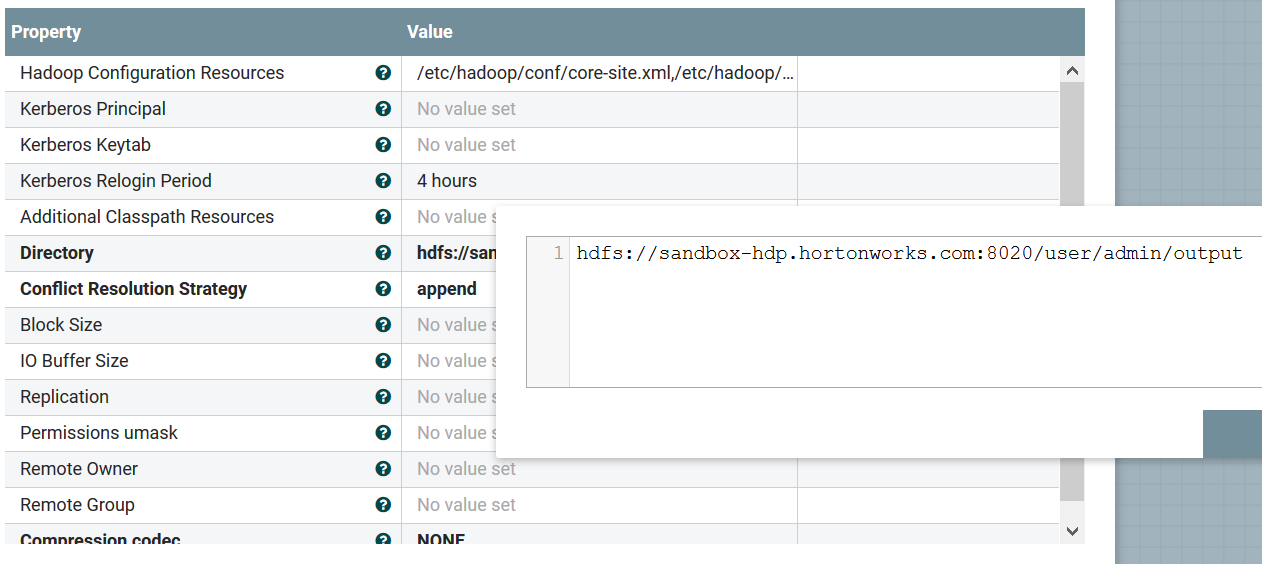


Figure : PutHDFS processor properties

ReplaceText is used to reshape records into CQL insert queries before sinking them using PutCassandraQL. It uses a regex expression to extract tokens and accomplish this reshaping.

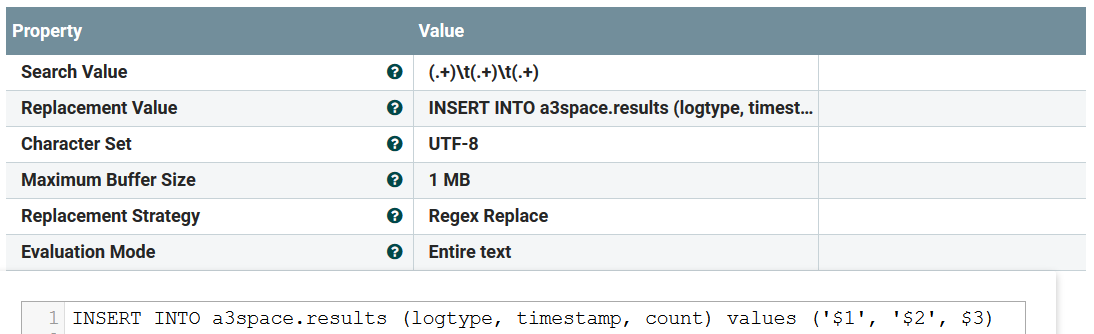


Figure : ReplaceText processor properties

PutCassandraQL is used to insert records into a Cassandra KeySpace and Table. It is an endpoint of the pipeline and discards all records after insertion.

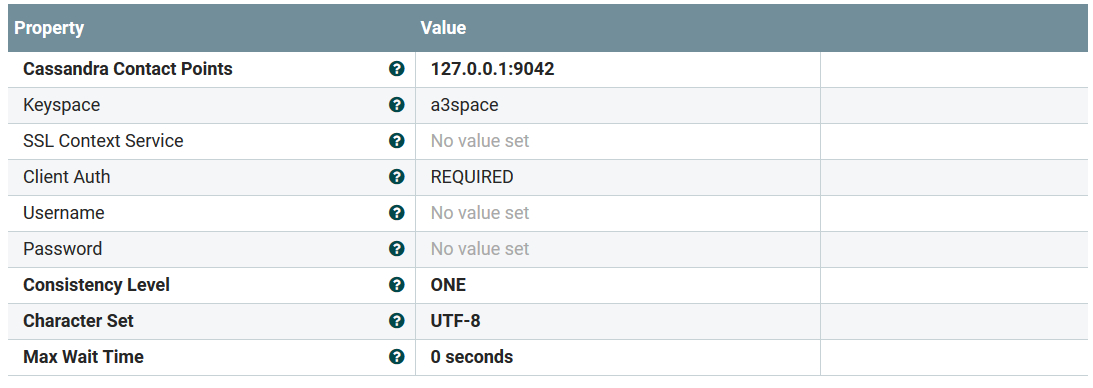


Figure : PutCassandraQL processor properties

The Nifi configuration was exported as a template in the file “ETL\_pipeline.xml”.

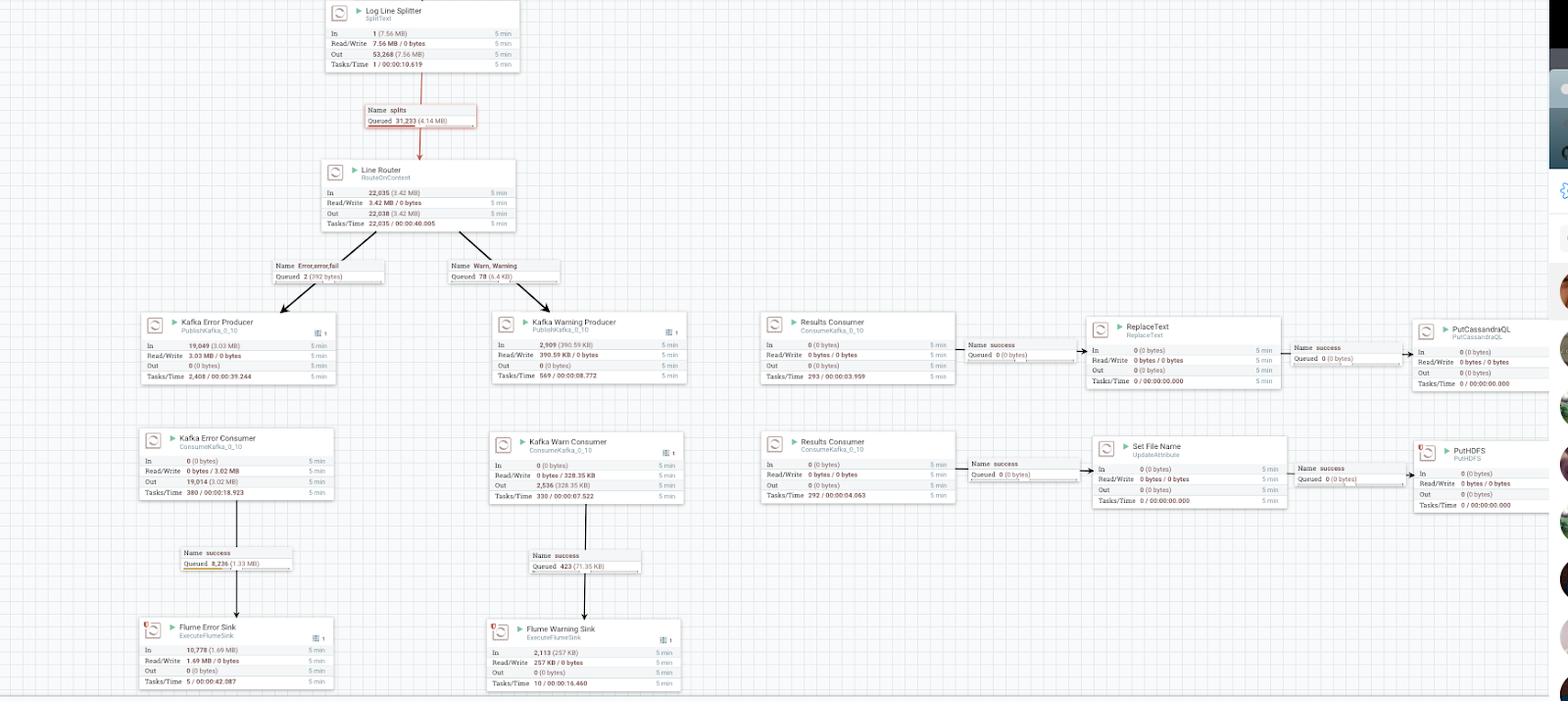


Figure : Overall Nifi Dataflow

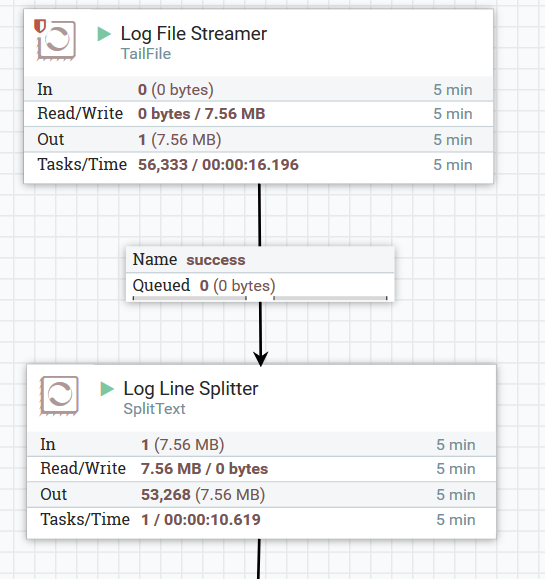


Figure : TailFile processor that did not fit in Figure 11

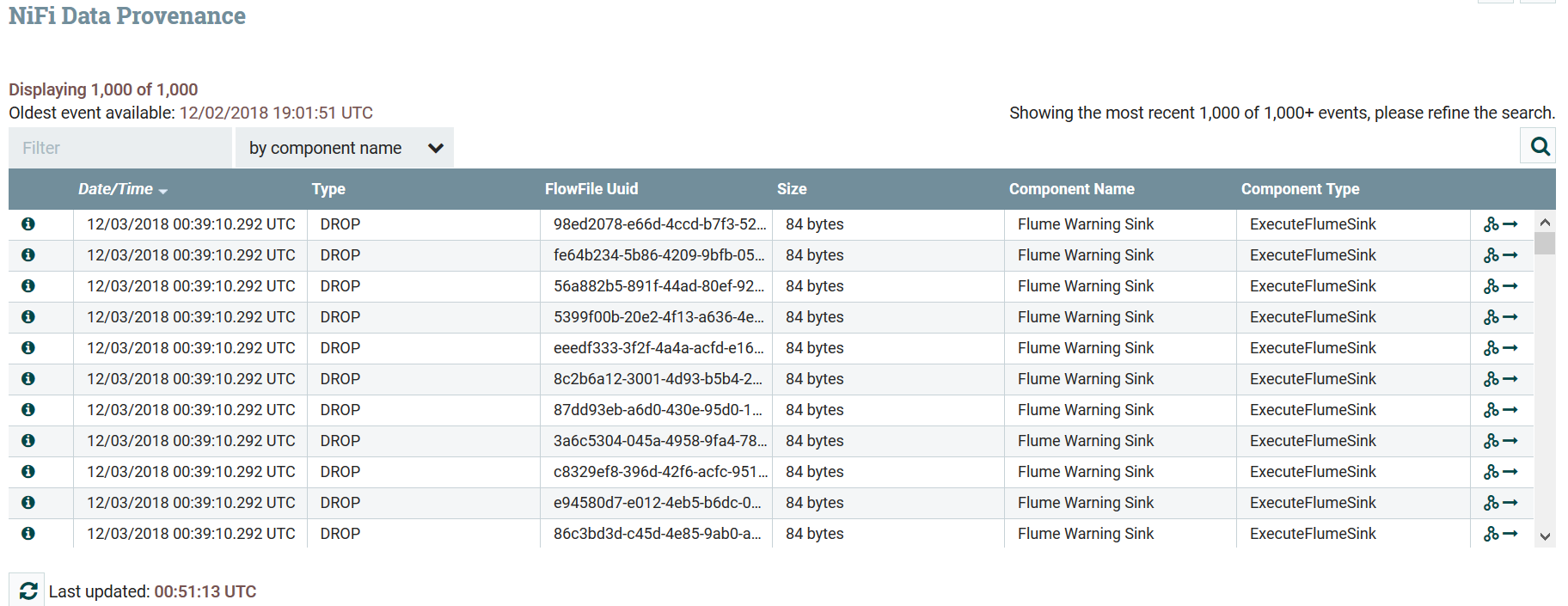


Figure : Nifi data provenance