KIP-307: Allow to define custom processor names with KStreams DSL

Status

Current state: *Accepted*

Discussion thread: *here*

JIRA: *here*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Currently, while building a new Topology through the KStreams DSL the processors are automatically named.

The genarated names are prefixed depending of the operation (i.e KSTREAM-SOURCE, KSTREAM-FILTER, KSTREAM-MAP, etc). Moreover, they are added with a suffix like "-0000000000" to guarantee their uniqueness.

To debug/understand a topology it is possible to display the processor lineage with the method Topology#describe(). However, a complex topology with dozens of operations can be hard to understand if the processor names are not relevant.

So allowing users to be able to set more meaningful names can help to resolve complexity of some developed topologies. For example, a processor name could describe the business rule performed by a map() operation.

Public Interfaces

First, we propose to add one new interface NamedOperation that can be used to customize stateless operations as well as stateful ones. The objective to create a new class is to keep consistent with the overall API design.

```
package org.apache.kafka.streams.kstream;

/**
 * Default interface which can be used to personalized the named of operations, internal topics or store.
 */
public interface NamedOperation<T extends NamedOperation<T>> {

    /**
     * Sets the name to be used for operation.
     *
     * @param name  the name to use.
     * @return an instance of {@link NamedOperation}
     */
    T withName(final String name);

}
```

The NamedOperation interface will be implemented/extended by following classes that already exist to configure operations :

- Produced
- Consumed

- Printed
- Joined
- Grouped
- Suppressed

In addition, we propose to add a new static method with the following signature to each of those class **as(final String processorName).**

Deprecration

In order to fix some inconsistencies in API, we also propose to deprecate the method **named(String)** from the class Joined in favor of new method **as()**.

In addition, as no configuration classes expose the processor name, we will also deprecate the method name() from the class Joined . This method should be removed in a future release.

Overloaded methods

Then, we propose to overload all stateless methods that do not have one of the existing control classes listed above to accept a NamedOperation implementation.

For that we will added a new default class Named implementing NamedOperation :

```
public class Named implements NamedOperation<Named> {

    private static final int MAX_NAME_LENGTH = 249;

    protected String name;

    protected Named(final String name) {
        this.name = name;
        if (name != null)
            validate(name);
    }

    /**
     * Create a Named instance with provided name.
     *
     * @param name  the processor name to be used. If {@code null} a default processor name will be generated.
     * @return      A new {@link Named} instance configured with name
     */
    public static Named as(final String name) {
        Objects.requireNonNull(name, "name can't be null");
        return new Named(name);
    }

    @Override
    public Named withName(final String name) {
        Objects.requireNonNull(name, "name can't be null");
        return new Named(name);
    }
...
}
```

The below tables will resume all new and existing methods :

KStream (16 new methods)

method	Added for this KIP ?	Object/method used for node name	Used for repartition topic name	Used for state store name ?
filter(Predicate, Named)	YES	static Named#as(String)	N/A	N/A
filterNot(Predicate, Named)	YES	static Named#as(String)	N/A	N/A
selectKey(KeyValueMapper, Named)	YES	static Named#as(String)	N/A	N/A
map(KeyValueMapper, Named)	YES	static Named#as(String)	N/A	N/A

mapValues(ValueMapper, Named)	YES	static Named#as(String)	N/A	N/A
mapValues(ValueMapperWithKey, Named)	YES	static Named#as(String)	N/A	N/A
flatMap(KeyValueMapper, Named)	YES	static Named#as(String)	N/A	N/A
flatMapValues(ValueMapper, Named)	YES	static Named#as(String)	N/A	N/A
flatMapValues(ValueMapperWithKey, Named)	YES	static Named#as(String)	N/A	N/A
print(Printed)	NO	static Printed#as(String)	N/A	N/A
foreach(ForeachAction, Named)	YES	static Named#as(String)	N/A	N/A
peek(ForeachAction, Named)	YES	static Named#as(String)	N/A	N/A
branch(Named, Predicate...)	YES	static Named#as(String)	N/A	N/A
through(String, Produced)	NO	static Produced#as(String)	N/A	N/A
to(String, Produced)	NO	static Produced#as(String)	N/A	N/A
to(TopicNameExtractor, Produced)	NO	static Produced#as(String)	N/A	N/A
transform(TransformerSupplier, Named, String...)	YES	static Named#as(String)	N/A	N/A
transformValues(ValueTransformerSupplier, Named, String...)	YES	static Named#as(String)	N/A	N/A
transformValues(ValueTransformerWithKeySupplier, Named, String...)	YES	static Named#as(String)	N/A	N/A
process(ProcessorSupplier, Named, String...)	YES	static Named#as(String)	N/A	N/A
join(KStream, ValueJoiner, JoinWindows windows, Joined)	NO	static Joined#as(final String name)	static Joined#as(final String name)	static Joined#as(final String name)
leftJoin(KStream, ValueJoiner, JoinWindows, Joined)	NO	static Joined#as(final String name)	static Joined#as(final String name)	static Joined#as(final String name)
outerJoin(KStream, ValueJoiner, JoinWindows, Joined)	NO	static Joined#as(final String name)	static Joined#as(final String name)	static Joined#as(final String name)
join(KTable, ValueJoiner, Joined)	NO	static Joined#as(final String name)	static Joined#as(final String name)	N/A
leftJoin(KTable, ValueJoiner, Joined)	NO	static Joined#as(final String name)	static Joined#as(final String name)	N/A
join(GlobalKTbale, KeyValueMapper, ValueJoiner, Named)	YES	static Named#as(String)	N/A	N/A
leftJoin(GlobalKTable, KeyValueMapper, ValueJoiner, Named)	YES	static Named#as(String)	N/A	N/A
flatTransform(TransformerSupplier, Named named, String... stateStoreNames)	YES	static Named#as(String)	N/A	N/A
flatTransformValues(ValueTransformerWithKeySupplier, Named, String...)	YES	static Named#as(String)	N/A	N/A
flatTransformValues(ValueTransformerSupplier, Named, String...)	YES	static Named#as(String)	N/A	N/A

KTable (16 new methods)

method	Added for this KIP ?	Object/method used for node name	Used for repartition topic name	Used for state store name ?
filter(Predicate, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
filter(Predicate, Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)
filterNot(Predicate, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
filterNot(Predicate, Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)
mapValues(ValueMapper, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
mapValues(ValueMapperWithKey, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
mapValues(ValueMapper, Named, Materialized)	YES	static Named#as(String)	N/A	static Materialized#as (String)
mapValues(ValueMapperWithKey, Named, Materialized);	YES	static Named#as(String)	N/A	static Materialized#as (String)

suppress(Suppressed)	NO	Suppressed#withName (String)	N/A	N/A
transformValues(ValueTransformerWithKeySupplier, Named, String...)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
transformValues(ValueTransformerWithKeySupplier, Materialized, Named, String...)	YES	static Named#as(String)	N/A	static Materialized#as (String)
groupBy(KeyValueMapper, KeyValue, Grouped)	NO	static Grouped#as(String)	static Grouped#as(String)	N/A
join(KTable, ValueJoiner, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
join(KTable, ValueJoiner, Named, Materialized)	YES	static Named#as(String)	N/A	static Materialized#as (String)
leftJoin(KTable, ValueJoiner, Named);	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
leftJoin(KTable, ValueJoiner, Named, Materialized)	YES	static Named#as(String)	N/A	static Materialized#as (String)
outerJoin(KTable, ValueJoiner, Named);	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
outerJoin(KTable, ValueJoiner, Named, Materialized)	YES	static Named#as(String)	N/A	static Materialized#as (String)
toStream(Named)	YES	static Named#as(String)	N/A	N/A
toStream(KeyValueMapper, Named)	YES	static Named#as(String)	N/A	N/A

KGroupedStream (6 new methods)

method	Added for this KIP ?	Object/method used for node name	Used for repartition topic name	Used for state store name ?
count(Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
count(Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)
reduce(Reducer, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
reduce(Reducer, Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)
aggregate(Initializer, Aggregator, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
aggregate(Initializer, Aggregator, Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)

KGroupedTable (6 new methods)

method	Added for this KIP ?	Object/method used for node name	Used for repartition topic name	Used for state store name ?
count(Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
count(Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)
reduce(Reducer, Reducer, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
reduce(Reducer, Reducer, Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)
aggregate(Initializer, Aggregator, Aggregator, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
aggregate(Initializer, Aggregator, Aggregator, Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)

TimeWindowedKStream (6 new methods)

method	Added for this KIP ?	Object/method used for node name	Used for repartition topic name	Used for state store name ?
count(Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
count(Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)
aggregate(Initializer, Aggregator, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
aggregate(Initializer, Aggregator, Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)

reduce(Reducer, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
reduce(Reducer, Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)

SessionWindowedKStream (6 new methods)

method	Added for this KIP ?	Object/method used for node name	Used for repartition topic name	Used for state store name ?
count(Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
count(Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)
aggregate(Initializer, Aggregator, Merger, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
aggregate(Initializer, Aggregator, Merger, Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)
reduce(Reducer, Named)	YES	static Named#as(String)	N/A	(PREFIX + COUNT)
reduce(Reducer, Named, Materialized)	YES	static Named#as(String)	N/A	Materialized#as(String)

At the end, we can summarize the scope of each configuration class as follow :

	Generated	Named	Joined / Grouped / Produced / Consumed	Materialized
Node Name	X	X	X	
Repartition Topic	X		X	
Queryable Store				X
State store	X		X	X
Changelog Topic	X		X	X

Materialized

The main reason why we propose to overload each method accepting a Materialized argument is to not introduce ambitguity by conflating config objects that configure an operation (like Grouped, Joined) with config objects that configure an aspect of the operation (like Materialized).

Name Validation

User provided node name should follow the same restrictions that ones currently apply to state stores during the create of Materialized instance.

Currently, the Materialized class relies on the static method Topic#validate. This method ensure that a provided name only contains legal characters [a-zA-Z0-9._-] and have a maximum length of 249.

We propose to copy methods from Topic#validate into Named. This new method will be used validate both store names and node names. The benefit is to remove a dependency with the core module.

In addition, the Materialized class will throw a TopologyException while building the topology in case of a unvalid name instead of InvalidTopicException .

Proposed Changes

- Implement the new interface NamedOperation and default class Named
- Update all parameter class to implement NamedOperation : Produced Consumed Printed Joined Grouped Suppressed
- Overload methods stateless for classes KStreams, KTables, KGroupedStream, KGroupedTable, TimeWindowedKStream, TimeWindowedKTable
- The processor names specified by developer will be used in place of the static processor prefix. Statics prefixes will still be used if no custom processor name are specified.
- Processor names should follow the same restrictions as the topic names. So legal characters are [a-zA-Z0-9._-] and the maximum length of 249.

Below is an application example :

```
final StreamsBuilder builder = new StreamsBuilder();


builder.stream("topic-input", Consumed.as("STREAM-FROM-TOPIC-INPUT")
        .filter((k, v) -> true ), Named.as("FILTER-NULL-VALUE")
        .map((k, v) -> KeyValue.pair(k, v.toUpperCase()), Named.as("MAP-TO-UPPERCASE")
        .to("topic-output", Produced.as("TO-OUTPUT-TOPIC"));

System.out.println(builder.build().describe());
---- (output)----
Topologies:
   Sub-topology: 0
    Source: STREAM-FROM-TOPIC-INPUT (topics: [topic-input])
      --> FILTER-NULL-VALUE
    Processor: FILTER-NULL-VALUE (stores: [])
      --> MAP-TO-UPPERCASE
      <-- STREAM-FROM-TOPIC-INPUT
    Processor: MAP-TO-UPPERCASE (stores: [])
      --> TO-OUTPUT-TOPIC
      <-- FILTER-NULL-VALUE
    Sink: TO-OUTPUT-TOPIC (topic: topic-output)
      <-- MAP-TO-UPPERCASE
```

A straightforward first pass is GitHub PR 6958

Compatibility, Deprecation, and Migration Plan

No compatibility issues foreseen.

Rejected Alternatives

1. *The first proposition was to add new methods KStreams#as(Described) and KTable#as(Described) while Described class would be used to customized the named of operation defined previously in the stream. However not only this new method was not conservative with the existing APIs but it also introduce some complexities for methods returning Void.*
2. *The second proposition was to enrich all actions classes (Reducer, Predicate, etc) with a new default method "as(String)" in order to name the operation. But this leads to mix different classes with different semantics (Predicate vs Consumed/Produced) creating a couple of unfortunate side-effects:.*