KIP 230: Name Windowing Joins

blocked URL

Status

Current state: *Discarded* (subsumed via KIP-372: Naming Repartition Topics for Joins and Grouping)

Discussion thread: here [Change the link from the KIP proposal email archive to your own email thread]

JIRA: here

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

Motivation

Currently, the internal name of a windowing join is generated at runtime using a counter to ensure uniqueness within a running topology. When a topology is changed and redeployed, different names can be generated for the same window. This results in lost windowing state without intervention modifying offsets to re-consume the input topics.

Public Interfaces

- https://kafka.apache.org/10/javadoc/org/apache/kafka/streams/kstream/Joined.html

Proposed Changes

The proposed change would be to

- Add an optional configuration parameter "joinName" to `org.apache.kafka.streams.kstream.Joined`.
- If provided, this would be used by KStreamImpl to generate internal names when building a join.
    - Windowing joins: The joinName would be used in the naming of the internal topics
    - Kstream-KTable joins: The joinName would be used in the naming of repartition topics, if they are needed.
- If not provided the existing name generation would be used.
- If a naming conflict occurs, generate a Topology Builder exception. This matches the behavior of existing methods that accept user-provided names, such as `reduce`:
    - org.apache.kafka.streams.errors.TopologyBuilderException: "Invalid topology building: Topic reduction-same-name-repartition has already been registered by another source."

Example changes:

JoinName Option added to Joined:

Joined

```
diff --git a/streams/src/main/java/org/apache/kafka/streams/kstream/Joined.java b/streams/src/main/java/org
/apache/kafka/streams/kstream/Joined.java
index 8601e1c6a..c5e892a09 100644
--- a/streams/src/main/java/org/apache/kafka/streams/kstream/Joined.java
+++ b/streams/src/main/java/org/apache/kafka/streams/kstream/Joined.java
@@ -27,13 +27,36 @@ public class Joined<K, V, VO> {
 private Serde<K> keySerde;
 private Serde<V> valueSerde;
 private Serde<VO> otherValueSerde;
+ private String joinName;

 private Joined(final Serde<K> keySerde,
 final Serde<V> valueSerde,
- final Serde<VO> otherValueSerde) {
+ final Serde<VO> otherValueSerde,
+ final String joinName) {
 this.keySerde = keySerde;
 this.valueSerde = valueSerde;
 this.otherValueSerde = otherValueSerde;
```

```
+ this.joinName = joinName;
+ }
+
+ /**
+ * Create an instance of {@code Joined} with key, value, and otherValue {@link Serde} instances.
+ * {@code null} values are accepted and will be replaced by the default serdes as defined in config.
+ *
+ * @param keySerde the key serde to use. If {@code null} the default key serde from config will be used
+ * @param valueSerde the value serde to use. If {@code null} the default value serde from config will be used
+ * @param otherValueSerde the otherValue serde to use. If {@code null} the default value serde from config
will be used
+ * @param <K> key type
+ * @param <V> value type
+ * @param <VO> other value type
+ * @param joinName name of the join used to generate backing topic names
+ * @return new {@code Joined} instance with the provided serdes
+ */
+ public static <K, V, VO> Joined<K, V, VO> with(final Serde<K> keySerde,
+ final Serde<V> valueSerde,
+ final Serde<VO> otherValueSerde,
+ final String joinName) {
+ return new Joined<>(keySerde, valueSerde, otherValueSerde, joinName);
 }

 /**
@@ -51,7 +74,7 @@ public class Joined<K, V, VO> {
 public static <K, V, VO> Joined<K, V, VO> with(final Serde<K> keySerde,
 final Serde<V> valueSerde,
 final Serde<VO> otherValueSerde) {
- return new Joined<>(keySerde, valueSerde, otherValueSerde);
+ return with(keySerde, valueSerde, otherValueSerde, null);
 }

 /**
@@ -96,6 +119,20 @@ public class Joined<K, V, VO> {
 return with(null, null, otherValueSerde);
 }

+ /**
+ * Create an instance of {@code Joined} with an other value {@link Serde}.
+ * {@code null} values are accepted and will be replaced by the default value serde as defined in config.
+ *
+ * @param joinName the joinName to use. If {@code null} the default value serde from config will be used
+ * @param <K> key type
+ * @param <V> value type
+ * @param <VO> other value type
+ * @return new {@code Joined} instance configured with the joinName
+ */
+ public static <K, V, VO> Joined<K, V, VO> otherValueSerde(final String joinName) {
+ return with(null, null, null, joinName);
+ }
+
 /**
 * Set the key {@link Serde} to be used. Null values are accepted and will be replaced by the default
 * key serde as defined in config
@@ -132,6 +169,18 @@ public class Joined<K, V, VO> {
 return this;
 }

+ /**
+ * Set the joinName to be used. Null values are accepted and will be replaced by the default
+ * value serde as defined in config
+ *
+ * @param joinName the joinName to use. If null the default joinName from config will be used
+ * @return this
+ */
+ public Joined<K, V, VO> withJoinName(final String joinName) {
+ this.joinName = joinName;
+ return this;
+ }
+
```

```
 public Serde<K> keySerde() {
 return keySerde;
 }
@@ -143,4 +192,8 @@ public class Joined<K, V, VO> {
 public Serde<VO> otherValueSerde() {
 return otherValueSerde;
 }
+
+ public String joinName() {
+ return joinName;
+ }
 }
```

JoinName used to generate internal topic name, if provided:

KStreamImpl.java

```
diff --git a/streams/src/main/java/org/apache/kafka/streams/kstream/internals/KStreamImpl.java b/streams/src
/main/java/org/apache/kafka/streams/kstream/internals/KStreamImpl.java
index 8e80315fa..8f9df8c19 100644
--- a/streams/src/main/java/org/apache/kafka/streams/kstream/internals/KStreamImpl.java
+++ b/streams/src/main/java/org/apache/kafka/streams/kstream/internals/KStreamImpl.java
@@ -362,12 +362,12 @@ public class KStreamImpl<K, V> extends AbstractStream<K> implements KStream<K, V
 return branchChildren;
 }

- @Override
+ @Override
 public KStream<K, V> merge(final KStream<K, V> stream) {
 Objects.requireNonNull(stream);
 return merge(builder, stream);
 }
-
+
 private KStream<K, V> merge(final InternalStreamsBuilder builder,
 final KStream<K, V> stream) {
 KStreamImpl<K, V> streamImpl = (KStreamImpl<K, V>) stream;
@@ -880,6 +880,10 @@ public class KStreamImpl<K, V> extends AbstractStream<K> implements KStream<K, V
 this.rightOuter = rightOuter;
 }

+ private <K1, V1, V2> String newProcessorName(final String prefix, final Joined<K1, V1, V2> joined) {
+ return joined.joinName() != null ? (prefix + joined.joinName()) : builder.newProcessorName(prefix);
+ }
+
 @SuppressWarnings("unchecked")
 public <K1, R, V1, V2> KStream<K1, R> join(final KStream<K1, V1> lhs,
 final KStream<K1, V2> other,
@@ -888,8 +892,8 @@ public class KStreamImpl<K, V> extends AbstractStream<K> implements KStream<K, V
 final Joined<K1, V1, V2> joined) {
 String thisWindowStreamName = builder.newProcessorName(WINDOWED_NAME);
 String otherWindowStreamName = builder.newProcessorName(WINDOWED_NAME);
- String joinThisName = rightOuter ? builder.newProcessorName(OUTERTHIS_NAME) : builder.newProcessorName
(JOINTHIS_NAME);
- String joinOtherName = leftOuter ? builder.newProcessorName(OUTEROTHER_NAME) : builder.newProcessorName
(JOINOTHER_NAME);
+ String joinThisName = rightOuter ? newProcessorName(OUTERTHIS_NAME, joined) : newProcessorName(JOINTHIS_NAME,
joined);
+ String joinOtherName = leftOuter ? newProcessorName(OUTEROTHER_NAME, joined) : newProcessorName
(JOINOTHER_NAME, joined);
 String joinMergeName = builder.newProcessorName(MERGE_NAME);

 final StoreBuilder<WindowStore<K1, V1>> thisWindow =
```

Compatibility, Deprecation, and Migration Plan

1. Since this is an opt-in, optional parameter there would be no impact on existing code.
2. The join builder methods already accept a Joined instance for configuration, so no API changes are needed in the builder.