

# KIP-698: Add Explicit User Initialization of Broker-side State to Kafka Streams


- [Status](#)
- [Motivation](#)
- [Public Interfaces](#)
- [Proposed Changes](#)
- [Compatibility, Deprecation, and Migration Plan](#)
- [Rejected Alternatives](#)

## Status

**Current state:** Accepted

**Discussion thread:** <https://lists.apache.org/x/thread.html/rf65435729171c40a001a394e9f6170e7a2c24b6c9424a346954cd0e1@%3Cdev.kafka.apache.org%3E>

JIRA:

 Unable to render Jira issues macro, execution error.

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

## Motivation

Kafka Streams uses repartition topics to repartition the data when a key-based operation, e.g., aggregation or join, follows a key-changing operation, e.g., map. Additionally, Kafka Streams uses changelog topics to replicate the data in its state store for fault-tolerance. If any required repartition topic or changelog topic does not exist, it is created during a rebalance. More precisely these internal topics are created during the computation of the assignment. Consequently, if one of the internal topics is deleted between rebalances, it will be silently recreated as an empty topic and data might be lost. Kafka Streams users would either not notice the loss at all or notice it too late to limit the damage by stopping processing. Deletion of internal topics may happen by mistake. This silent recreation of internal topics could be avoided by creating the internal topics only once during the first-ever rebalance of the application (or after an application reset). However, determining the first-ever rebalance of an application is not always straightforward. For example, if all Kafka Streams clients of a Kafka Streams application are stopped, an internal topic is deleted, and then the Kafka Streams application is restarted, the deleted topic would be silently recreated as an empty topic because Kafka Streams does not have any information that it could leverage to recognize that the first rebalance after restarting is not the first-ever rebalance of the application.

We propose to move the creation of the internal topics to an initialization method that users can call before they start their application for the first time. That way, users have full control over when the internal topics are created and Kafka Streams can reliably notify the users about possible data loss by throwing an exception when internal topics do unexpectedly not exist or are misconfigured. Additionally, we propose to add a configuration that allows to turn automatic creation of internal topics on or off. The configuration is needed for backward compatibility but also to keep the first steps with Kafka Streams simple for new users.

## Public Interfaces

We propose to add two new exceptions `MissingInternalTopicsException` and `MisconfiguredInternalTopicException`. `MissingInternalTopicsException` is thrown when a required internal topic does not exist and `MisconfiguredInternalTopicException` is thrown when the internal topic is not configured as expected. In general, misconfigurations are configurations that differ from default values specified by Kafka Streams or from configurations specified in user code where configurations in user code are considered before Streams' default values. Those misconfigurations typically occur if users change configurations directly on the brokers. For example, a changelog topic for a non-windowed state store is regarded as misconfigured if its retention period is not set to unlimited. Another example for a misconfigured internal topic is a repartition topic with the wrong number of partitions. In future, we might discover other misconfigurations that are critical to data loss. In such a case, Kafka Streams will also throw a `MisconfiguredInternalTopicException` without the need of a new KIP. The exception will contain a detailed explanation of what is wrong and how to fix it. The intended recipient of these exceptions is the operator, not software.

These exceptions help to discriminate errors originating from missing or misconfigured internal topics from other errors in the uncaught exception handler. For example, reacting on a missing source topic (i.e., `MissingSourceTopicException`) might be different from reacting on a missing or misconfigured internal topic, because the process for re-creating source topics might differ from the process for re-creating internal topics. Furthermore, source topics might be owned by a different team than the internal topics, consequently different people need to be paged. Exceptions `MissingInternalTopicsException` and `MisconfiguredInternalTopicException` will be thrown during explicit initialization and during a rebalance when the internal topics are verified. Both exceptions are fatal.

Additionally, we propose to add an exception `InternalTopicsAlreadySetupException` that is thrown if users attempt to initialize an already initialized application. Exception `InternalTopicsAlreadySetupException` will be only thrown during explicit initialization. Such a behavior will ensure that users explicitly initialize the application only when:

- they start the application for the first time,
- they restart the application after an application reset,
- they want to create missing internal topics of an incomplete setup (by setting the parameters of the initialization appropriately as described below).

If we do not throw exception `InternalTopicsAlreadySetupException`, users might be tempted to initialize the application at each restart of the application. The initialization would not throw if all internal topics exist and it would only throw if some but not all internal topics were missing. If all internal topics were accidentally deleted between two restarts, the initialization would silently recreate all internal topics which might again lead to data loss. `InternalTopicsAlreadySetupException` is fatal.

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

public class MissingInternalTopicsException extends StreamsException {
    public List<String> topics();
}

public class MisconfiguredInternalTopicException extends StreamsException {
}

public class InternalTopicsAlreadySetupException extends StreamsException {
}
```

We propose to add an initialization method to the `KafkaStreams` class.

```

public class KafkaStreams {

    /**
     * Initializes broker-side state.
     *
     * @throws MissingSourceTopicException if a source topic is missing
     * @throws MissingInternalTopicsException if some but not all of the internal topics are missing
     * @throws MisconfiguredInternalTopicException if an internal topics is misconfigured
     * @throws InternalTopicsAlreadySetupException if all internal topics are already setup
     */
    public void init();

    /**
     * Initializes broker-side state.
     *
     * @throws MissingSourceTopicException if a source topic is missing
     * @throws MissingInternalTopicsException if some but not all of the internal topics are missing
     * @throws MisconfiguredInternalTopicException if an internal topics is misconfigured
     * @throws InternalTopicsAlreadySetupException if all internal topics are already setup
     * @throws TimeoutException if initialization exceeds the given timeout
     */
    public void init(final Duration timeout);

    /**
     * Initializes broker-side state.
     *
     * This methods takes parameters that specify which internal topics to setup if some
     * but not all of them are absent.
     *
     * @throws MissingSourceTopicException if a source topic is missing
     * @throws MissingInternalTopicsException if some but not all of the internal topics are missing
     * and the given initialization parameters do not specify to
    setup them
     * @throws MisconfiguredInternalTopicException if an internal topics is misconfigured
     * @throws InternalTopicsAlreadySetupException if all internal topics are already setup
     */
    public void init(final InitParameters initParameters);

    /**
     * Initializes broker-side state.
     *
     * This methods takes parameters that specify which internal topics to setup if some
     * but not all of them are absent.
     *
     * @throws MissingSourceTopicException if a source topic is missing
     * @throws MissingInternalTopicsException if some but not all of the internal topics are missing
     * and the given initialization parameters do not specify to
    setup them
     * @throws MisconfiguredInternalTopicException if an internal topics is misconfigured
     * @throws InternalTopicsAlreadySetupException if all internal topics are already setup
     * @throws TimeoutException if initialization exceeds the given timeout
     */
    public void init(final InitParameters initParameters, final Duration timeout);

    public class InitParameters {

        public static InitParameters initParameters(); // specifies to disable the setup
        of internal topics if some topics are missing

        public InitParameters enableSetupInternalTopicsIfIncomplete(); // specifies to setup
        repartition and changelog topics if some are missing
        public InitParameters disableSetupInternalTopicsIfIncomplete(); // specifies to throw if
        some but not all repartition or changelog topics are missing
        public boolean setupInternalTopicsIfIncompleteEnabled(); // getter
    }
}

```

We propose to add a new configuration to Kafka Streams to determine whether the internal topics should be setup during a rebalance or by users calling `KafkaStreams#init()`.

```
public class StreamsConfig {

    // possible values
    public static final String AUTOMATIC_SETUP = "automatic";
    public static final String MANUAL_SETUP = "manual";

    // configuration
    public static final String INTERNAL_TOPIC_SETUP = "internal.topics.setup"; // default is AUTOMATIC_SETUP
}
```

## Proposed Changes

If configuration `INTERNAL_TOPIC_SETUP` is set to `AUTOMATIC_SETUP`, the group leader will set up the internal topics during a rebalance. If internal topics were deleted between rebalances, the group leader will create the deleted internal topics during the rebalance. That corresponds to the current behavior of Kafka Streams. Users can also call `KafkaStreams#init()` to set up the internal topics when `INTERNAL_TOPIC_SETUP` is set to `AUTOMATIC_SETUP`, but the call is not necessary since the internal topics would be created anyways during the next rebalance. Additionally, misconfigurations will be automatically rectified if possible and/or logged.

If configuration `INTERNAL_TOPIC_SETUP` is set to `MANUAL_SETUP`, the group leader will not set up internal topics during a rebalance but users need to call `KafkaStreams#init()` to set up the internal topics. If the internal topics do not exist during a rebalance because `KafkaStreams#init()` has not been called or one or more internal topics have been deleted, a `MissingInternalTopicsException` is thrown in each Kafka Streams client. If during a rebalance the group leader realizes that an internal topic is misconfigured, a `MisconfiguredInternalTopicException` is thrown in each Kafka Streams client.

If method `KafkaStreams#init()`:

- does not find any internal topic for the Kafka Streams client on the brokers, it will setup all internal topics
- finds all internal topics for the Kafka Streams client on the brokers, it will throw `InternalTopicsAlreadySetupException`
- finds some of the internal topics it will
  - throw `MissingInternalTopicsException` if `KafkaStreams#init()` is called without any parameters or the parameters specify to throw in case of missing internal topics
  - setup the missing internal topics if the parameters passed to `KafkaStreams#init()` specify so
- finds a misconfigured internal topic it will throw `MisconfiguredInternalTopicException`
- does not complete within the given timeout it will throw `TimeoutException`
- does not find a source topic, it will throw `MissingSourceTopicException`

In addition to setup internal topics, `KafkaStreams#init()` will make all checks that are currently done during a rebalance including checks for source topics.

## Compatibility, Deprecation, and Migration Plan

Since we introduce configuration `INTERNAL_TOPIC_SETUP` with default value `AUTOMATIC_SETUP` that ensures the current Kafka Streams behavior, this KIP should not affect backward-compatibility.

We do not need to deprecate any public interfaces since we propose to add a new method to the public API that does not replace any other method.

Migrating from the current behavior to `INTERNAL_TOPIC_SETUP` set to `MANUAL_SETUP` can be done without any specific migration plan. Users need to set `INTERNAL_TOPIC_SETUP` to `MANUAL_SETUP` and need to change their code to call `KafkaStreams#init()` accordingly.

## Rejected Alternatives

- **Persist a flag for the first-ever rebalance broker side:** This approach was rejected because that would imply changes on the brokers which we thought we can avoid and still get a good solution with the approach proposed in this KIP.
- **Use committed offsets for a repartition topic to verify if a repartition topic existed:** This approach would not work since committed offsets are removed when a topic is deleted.