

KIP-793: Allow sink connectors to be used with topic-mutating SMTs

- [Status](#)
- [Motivation](#)
- [Public Interfaces](#)
 - [org.apache.kafka.connect.sink.SinkRecord](#)
- [Proposed Changes](#)
- [Compatibility, Deprecation, and Migration Plan](#)
 - [Old / existing sink connector plugins running on a new Kafka Connect version](#)
 - [New / updated sink connector plugins running on an older Kafka Connect version](#)
- [Rejected Alternatives](#)


Status

Current state: "Accepted"

Discussion thread: [here](#)

Vote thread: [here](#)

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

The Kafka Connect framework allows sink connector tasks to do their own offset tracking in case they want to do asynchronous processing (for instance, buffering records sent by the framework to be flushed to the sink system at some later time). The `SinkTask::preCommit` method allows sink task implementations to provide the framework with the consumer offsets that are safe to commit for each topic partition. There's currently an incompatibility between Sink connectors [overriding](#) the `SinkTask::preCommit` method and SMTs that mutate the topic field of a `SinkRecord` (for instance, the `RegexRouter` SMT that ships with Apache Kafka).

This problem has been present since the `SinkTask::preCommit` method's inception and is rooted in a mismatch between the Kafka topic partition and offset that is passed to `SinkTask::open` / `SinkTask::preCommit` (the original topic partition and offset before any transformations are applied) and the topic/partition/offset that is present in the `SinkRecord` that the `SinkTask::put` method receives (after transformations are applied). Since that's all the information the connector has to implement any kind of internal offset tracking, the topic/partition/offset it can return in `preCommit` will correspond to the transformed topic/partition/offset, when the framework actually expects it to be the original topic/partition/offset.

In

 Unable to render Jira issues macro, execution error.

the problem was fixed for connectors that don't override `SinkTask::`

`preCommit`. For the others, it was acknowledged that "broader API changes are required".

Public Interfaces

`org.apache.kafka.connect.sink.SinkRecord`

Add new fields corresponding to a record's pre-transform ("original") topic / partition / offset along with their corresponding getters and constructor:

SinkRecord

```
private final String originalTopic;
private final Integer originalKafkaPartition;
private final long originalKafkaOffset;
...
public SinkRecord(String topic, int partition, Schema keySchema, Object key, Schema valueSchema, Object
value, long kafkaOffset,
                  Long timestamp, TimestampType timestampType, Iterable<Header> headers, String
originalTopic, Integer originalKafkaPartition, long originalKafkaOffset)
...

/**
 * @return the topic corresponding to the Kafka record before any transformations were applied. This should
be
 * used for any internal offset tracking purposes rather than {@link #topic()}, in order to be compatible
 * with SMTs that mutate the topic name.
 */
public String originalTopic() {
    return originalTopic;
}

/**
 * @return the topic partition corresponding to the Kafka record before any transformations were applied.
This
 * should be used for any internal offset tracking purposes rather than {@link #kafkaPartition()}, in order
to be
 * compatible with SMTs that mutate the topic partition.
 */
public Integer originalKafkaPartition() {
    return originalKafkaPartition;
}

/**
 * @return the offset corresponding to the Kafka record before any transformations were applied. This
be
 * should be used for any internal offset tracking purposes rather than {@link #kafkaOffset()}, in order to
 * compatible with SMTs that mutate the offset value.
 */
public long originalKafkaOffset() {
    return originalKafkaOffset;
}
```

Proposed Changes

Expose the original (pre-transformation) Kafka topic, topic partition and offset via new `SinkRecord` public methods and ask sink task implementations to use that information for offset tracking purposes (i.e. the topic partition offsets that the sink task returns to the framework via `SinkTask::preCommit`). Note that while the record's offset can't be modified via the standard `SinkRecord::newRecord` methods that SMTs are expected to use, `SinkRecord` has public constructors that would allow SMTs to return records with modified offsets. This is why the proposed changes include a new `SinkRecord::originalKafkaOffset` method as well.

Compatibility, Deprecation, and Migration Plan

Old / existing sink connector plugins running on a new Kafka Connect version

Old / existing sink connector plugins will continue running as expected on new versions of Kafka Connect but they will continue to remain incompatible with topic-mutating SMTs.

New / updated sink connector plugins running on an older Kafka Connect version

To ensure that connectors using the new `SinkRecord` getter methods can still be deployed on older versions of Kafka Connect, a try catch block should be used to catch and handle `NoSuchMethodError` / `NoClassDefFoundError` (also see [KIP-610: Error Reporting in Sink Connectors](#) for reference which had the same issue). However, it should be clearly documented that the connector won't be compatible with topic/partition/offset mutating SMTs when deployed on older versions of Kafka Connect.

Rejected Alternatives

1. Address the offsets problem entirely within the framework, doing some kind of mapping from the transformed topic back to the original topic.
 - This would only work in the cases where there's no overlap between the transformed topic names, but would break for the rest of the transformations (e.g. static transformation, topic = "a").
 - Even if we wanted to limit the support to those cases, it would require considerable bookkeeping to add a validation to verify that the transformation chain adheres to that expectation (and fail fast if it doesn't).
2. Expose the entire original record instead of only topic / partition / offset (e.g. `originalSinkRecord`)
 - We should not expose the original value / key, transformations might be editing them for security reasons.
3. Create a method in the `SinkTaskContext` to get the original topic / partition / offset information, instead of updating `SinkRecord` (e.g. `SinkTaskContext.getOriginalTopic(SinkRecord sr) / SinkTaskContext.getOriginalKafkaPartition(SinkRecord sr)`)
 - Requires extra bookkeeping without concrete value.
4. Update `SinkTask::put` in any way to pass the new information outside `SinkRecord` (e.g. a Map or a derived class)
 - Much more disruptive change without considerable pros
5. Use record headers to expose the original topic / partition / offset instead of adding new methods to `SinkRecord`
 - While this gets around the need to handle `NoSuchMethodError` / `NoClassDefFoundError` in sink connectors that want to retain compatibility with older Connect runtimes, it is arguably even less intuitive from a connector developer's standpoint.
6. Add new overloaded `SinkTask::open` and `SinkTask::close` methods which include both pre-transform and post-transform topic partitions so that connectors can create and destroy resources even if they choose to use post-transform topic partitions to write to the sink system
 - This requires sink connector plugins to implement two new additional methods without significant benefit. The information from the existing open / close methods on pre-transform topic partitions can be combined with the per record information of both pre-transform and post-transform topic partitions to handle most practical use cases without significantly muddying the sink connector related public interfaces.
7. Keep the existing `SinkTask::open` and `SinkTask::close` methods but call them with post-transform rather than pre-transform topic partitions
 - This would break backward compatibility since there could exist connector plugins that are aware of the current limitations (i.e. that `SinkTask::open` / `SinkTask::close` are called for pre-transform topic partitions but records in `SinkTask::put` only currently contain their post-transform topic partition) and work around them. Also, pre-transform / original topic partitions and offsets are critical for any sink connectors that wish to implement exactly-once semantics by tracking offsets in the sink system rather than Kafka.