KIP-204 : Adding records deletion operation to the new Admin Client API

Status

Current state: *"Accepted"*

Discussion thread: *here*

JIRA: *KAFKA-5925*

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

Motivation

The KIP-107 provides a way to delete messages starting from a specified offset inside a topic partition which we don't want to take anymore so without relying on time-based and size-based log retention policies. The already implemented protocol request and response messages (DeleteRecords API, key 21) are used only by the "legacy" Admin Client in Scala and aren't provided by the new Admin Client API in Java. This KIP is about adding this feature to the new Admin Client API.

Note that this KIP is related to KIP-107.

Public Interfaces

The AdminClient API will have new methods added (plus overloads for options):

```
deleteRecords(Map<TopicPartition, RecordsToDelete> recordsToDelete)
```

Proposed Changes

AdminClient : deleteRecords()

```
public DeleteRecordsResult deleteRecords(Map<TopicPartition, RecordsToDelete> recordsToDelete)
public DeleteRecordsResult deleteRecords(Map<TopicPartition, RecordsToDelete> recordsToDelete,
DeleteRecordsOptions options)
```

Where :

`TopicPartition` comes from `org.apache.kafka.common package`

`RecordsToDelete, DeleteRecordsOptions` and `DeleteRecordsResult` are defined as follow.

```
/**
 * Options for {@link AdminClient#deleteRecords(Map, DeleteRecordsOptions)}.
 */
public class DeleteRecordsOptions extends AbstractOptions<DeleteRecordsOptions> {

}

/**
 * Describe records to delete in a call to {@link AdminClient#deleteRecords(Map)}
 */
public class RecordsToDelete {
        private long offset;

        /**
         * Delete all the records before the given {@code offset}
         *
         * @param offset    the offset before which all records will be deleted
         */
        public static RecordsToDelete beforeOffset(long offset) { ... }
}

/**
 * The result of the {@link AdminClient#deleteRecords(Map)} call.
 */
public class DeleteRecordsResult {
    // package access constructor
    Map<TopicPartition, KafkaFuture<DeleteRecords>> values() { ... }
    KafkaFuture<DeleteRecords> all() { ... }
}

/**
 * Represents information about deleted records
 */
public class DeletedRecords {
        private final long lowWatermark;

        /**
          * Create an instance of this class with the provided parameters.
          *
         * @param lowWatermark  "low watermark" for the topic partition on which the deletion was executed
         */
        public DeletedRecords(long lowWatermark) {
            this.lowWatermark = lowWatermark;
        }

        /**
         * Return the "low watermark" for the topic partition on which the deletion was executed
         */
        public long lowWatermark() {
            return lowWatermark;
        }
}
```

In the `DeleteRecordsResult`, the Long value accessed by `values()` and `all()` method specifies the low watermark as described in the KIP-107.

The `deleteRecords()` will have the same authorization setting as `deleteTopic()`. Its operation type is be DELETE and its resource type is TOPIC.

Compatibility, Deprecation, and Migration Plan

This is a new API and won't directly affect existing users.

It should replace the deleteRecordsBefore() method provided by the Scala based Admin Client. So existing users who are using such method, should migrate to use the new Java based Admin Client for this feature.

Rejected Alternatives

If there are alternative ways of accomplishing the same thing, what were they? The purpose of this section is to motivate why the design is the way it is and not some other way.