

KIP-229: DeleteGroups API


- [Motivation](#)
- [Public Interfaces](#)
- [Proposed Changes](#)
- [Rejected Alternatives](#)

Status

Current state: [Accepted](#)

Discussion 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 old consumer based group management allows for explicitly deleting a consumer group. This functionality is missing in the new consumer based group management where offsets are stored in a Kafka internal topic. The only way a consumer group can be removed is when its offsets are expired. The default retention time for committed offsets is currently 1 day ([KIP-186](#) aims to increase that to 7 days). The override for this retention time is through the [offsetCommit API](#). Therefore, the offset cache can easily expand for consumer groups that are just waiting for their offsets to expire. With [KIP-211](#) that proposes removing the `retention_time` field from the `OffsetCommit` API, the need for an alternate group deletion mechanism is more eminent than ever.

To mimic the tooling behavior of the old consumer group tool, this KIP proposes supporting the `--delete` switch for Kafka stored group offsets (new consumer) too. The behavior of `--delete` for the new consumer will be similar to that of the old consumer.

Public Interfaces

While existing APIs remain unchanged a new API `DeleteGroups` is introduced to remove offsets of the given group(s) from the internal offset cache.

Proposed Changes

A new API is introduced for deleting consumer groups:

```
DeleteGroups Request (Version: 0) => [group_id]
  group_id => STRING

DeleteGroups Response (Version: 0) => throttle_time_ms [group_error_codes]
  throttle_time_ms => INT32
  group_error_codes => group_id error_code
    group_id => STRING
    error_code => INT16
```

Protocol	Field	Description
Request	group_id	The unique group identifier
Response	throttle_time_ms	Duration in milliseconds for which the request was throttled due to quota violation (Zero if the request did not violate any quota)
	group_error_codes	
	group_id	The unique group identifier

error_code	Error code corresponding to the individual group_id
------------	---

In addition to this new API, we define the necessary operation/resource permission to use the API as `Delete/Group`. In other words, when SASL authentication is enabled a user needs to have `Delete` operation access to a group in order to delete it. With this new `Delete` operation, the `Group` resource type will have four associated operations: `Describe`, `Read`, `Delete`, `All`. The `All` operation implies the other three, and the `Read` operation implies `Describe`. There is no *imply* relationship between `Delete` and `Read` or `Describe`.

Potential error codes returned by the API (at the top level or per-group level or both) include:

- 0: None
- 15: COORDINATOR_NOT_AVAILABLE
- 24: INVALID_GROUP_ID
- 30: GROUP_AUTHORIZATION_FAILED
- ??: NON_EMPTY_GROUP
- ?? : GROUP_ID_NOT_FOUND

Note that this KIP proposes two new error codes:

- NON_EMPTY_GROUP: addresses the case where the group is not empty (i.e. group state is not `Empty`) and cannot be deleted.
- GROUP_ID_NOT_FOUND: addresses the case where the requested group id does not exist.

A `--delete` switch for the consumer group command tool (`kafka-consumer-groups.sh`) will be supported for Kafka stored group offsets (new consumer) too, which behaves similar to its old consumer counterpart. For example, if group has active consumers the deletion will fail. In other words, a group can be deleted only when its state is `Empty`.

Compatibility, Deprecation, and Migration Plan

The changes proposed by this KIP apply to the new Java-based consumer only, and are backward compatible because the suggested API does not already exist. New clients that work with older brokers will not be able to make use of this new API.

Rejected Alternatives

1. Using the newly introduced offset reset functionality of the consumer group command tool by injecting a `retention_time` of 0: This alternative was rejected in favor of [KIP-211](#), that aims at removing the `retention_time` field from the `OffsetCommit` API.
2. Exposing the group deletion functionality in `org.apache.kafka.clients.admin.AdminClient`: This was left as a future work as no group-related functionality has been implemented in `AdminClient` yet.
3. Providing the functionality to delete individual offsets: This was also left as a potential future work to keep the initial implementation simple. If, in the future, it turns out that per-partition offset reset is not sufficient and there is a need to delete individual offsets, it can be proposed and implemented in its own KIP.
4. Having a high level error code in the response protocol: This was rejected because one error code cannot normally be generalized to all groups that are requested be deleted. It may very well be the case that a `Errors.COORDINATOR_NOT_AVAILABLE` error would apply to some groups (groups to be deleted may belong to different coordinators), `Errors.GROUP_AUTHORIZATION_FAILED` would apply to some other, etc.