KIP-547: Extend ConsumerInterceptor to allow modification of Consumer Commits

This page is meant as a template for writing a KIP. To create a KIP choose Tools->Copy on this page and modify with your content and replace the heading with the next KIP number and a description of your issue. Replace anything in italics with your own description.

Status

Current state: Under Discussion

Discussion thread: *here*

JIRA: TODO

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

Motivation

The CommitOffset api allows users to include a string of metadata while committing. This metadata can be useful for tracking information about the commit such as when and where it came from.

Unfortunately, the KafkaConsumer interface does not provide consistent access to set this metadata. The only way for a user to provide metadata for a commit is to manually construct the list of offsets to commit. The argument-less commit methods and auto-commit functionality do not expose any hooks to provide metadata and default to using an empty string. The inability to access the commit metadata for these methods makes it challenging to provide or enforce consistent metadata across multiple applications within in organization.

In addition, it would be useful for users to be able to manipulate or filter the actual offsets to be committed. An example would be filtering out commits that are identical to the most recent successful commit.

This KIP proposes an additional method in the ConsumerInterceptor that would allow for the modification of OffsetsAndMetadata to be committed to the cluster.

Public Interfaces

org.apache.kafka.clients.consumer.ConsumerInterceptor

```
 /**
  * This is called just before a commit request is sent to the cluster.
  * <p>
  * This method is allowed to modify the offsets that will be committed, in which case the new offsets will be
  * committed. There is no limitation on the number of offsets that can be returned from this method. I.e., the
  * interceptor can filter the offsets or generate new offsets.
  * <p>
  * Any exception thrown by this method will be caught by the caller, logged, but not propagated to the client.
  * <p>
  * Since the consumer may run multiple interceptors, a particular interceptor's preCommit() callback will
  * be called in the order specified by {@link org.apache.kafka.clients.consumer.
ConsumerConfig#INTERCEPTOR_CLASSES_CONFIG}.
  * The first interceptor in the list gets the offsets to commit, the following interceptor will be passed the
offsets
  * returned by the previous interceptor, and so on. Since interceptors are allowed to modify offsets,
interceptors
  * may potentially get offsets already modified by other interceptors. However, building a pipeline of mutable
  * interceptors that depend on the output of the previous interceptor is discouraged, because of potential
  * side-effects caused by interceptors potentially failing to modify the offsets and throwing an exception.
  * If one of the interceptors in the list throws an exception from preCommit(), the exception is caught,
logged,
  * and the next interceptor is called with the offsets returned by the last successful interceptor in the list,
  * or otherwise the original offsets to commit.
  *
  * @param offsets offsets to be committed to the cluster, or offsets returned by the previous interceptor in
the list
  * @return offsets that are either modified by the interceptor or the same offsets passed to this method.
  */
  public default Map<TopicPartition, OffsetAndMetadata> preCommit(Map<TopicPartition, OffsetAndMetadata>
offsets) {
    return offsets;
  }
```

Proposed Changes

This KIP proposes adding an additional method to the ConsumerInterceptor interface that can be used to manipulate both the offsets to be committed to a cluster as well as the metadata to be committed.

The new method will have similar behavior and caveats to the onConsume method. Specifically, The ability to freely modify the contents of the Map containing the offsets to commit, the ability to chain multiple calls together, and the suppression of exceptions.

An additional "preCommit" method will also be added to the internal ConsumerInterceptors object to handle the chaining of multiple interceptors. This method is basically identical to the existing onConsume method.

ConsumerInterceptors#preCommit() will be inserted as the first call in ConsumerCoordinator#sendOffsetCommitRequest(). It will be inserted prior to the check for an empty set of commits to allow Interceptors to remove all committed offsets from a request if they see fit.

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?*
  - No impact
  - The new method will be added with a default implementation that simply returns the input offsets
- *If we are changing behavior how will we phase out the older behavior?*
  - N/A
- *If we need special migration tools, describe them here.*
  - N/A
- *When will we remove the existing behavior?*
  - N/A

Rejected Alternatives

Adding a new interface specifically to manipulate commits was considered, but would require a significantly higher amount of code changes. The additional complexity to developers of discovering and learning about a new interface is similarly undesirable.

Future Improvements

The Kafka Protocol currently uses a single String for the commit metadata. This presents a potential challenge for developers using the preCommit method as they need to append their metadata to the end of the existing metadata string. It also presents a challenge for anyone who wants to do any kind of processing based on the metadata as the metadata string may have a mix of strange metadata that the processor would need to deal with. Adding Headers to the commit protocol and deprecating the current metadata string could reduce these challenges, but is outside the scope of this KIP.

The Producer also has the ability to commit offsets as part of the Transactional interface, however it currently does not have the ability to configure ConsumerInterceptors. As such it has never had access to the onCommit method of the ConsumerInterceptor and will not have access to the new preCommit method. Giving the producer access to these interceptors would bring further consistency to the commit interfaces, but would likely require a more complicated change than simply adding ConsumerInterceptors. Adding ConsumerInterceptors would likely add confusion, but more logical solutions such as splitting the commit methods into a separate CommitInterceptor interface would require a much more in-depth discussion. As such, these changes are outside of the scope of this KIP