KIP-799: Align behaviour for producer callbacks with documented behaviour

Status

Current state: *"Under Discussion"*

Discussion thread: here

JIRA: *here*

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

Motivation

In PR #4188 as part of KAFKA-6180, a breaking change was accidentally introduced in the behaviour of Callbacks for the producer client.

Previously, whenever an exception was thrown when producing an event, the value for 'metadata' passed to the Callback.onCompletion method was always null. In PR #4188 in one part of the code where Callback.onCompletion is called, the behaviour was changed so that instead of passing a null value for metadata, a 'placeholder' value was provided instead (see here and here). This placeholder contained only topic and partition information, and with all other fields set to '-1'.

This change only impacted a subset of exceptions, so that in the case of ApiExceptions metadata would still be null (see here), but for all other exceptions the placeholder value would be used. The behaviour at the time of writing remains the same.

This issue was first reported in KAFKA-7412 when a user first noticed the difference between the documented behaviour of Callback.onCompletion and the implemented behaviour.

At the time it was assumed that the behaviour when errors occur was to always provide a placeholder metadata value to Callback.onCompletion, and the documentation was updated at that time to reflect this assumption in PR #5798. The documentation now states that when an exception occurs that the method will be called with an empty metadata value (see here). However, there is still one case where Callback.onCompletion is called with a null value for metadata (see here), so there is still a discrepancy between the documented behaviour and the implementation of Callback.onCompletion.

Public Interfaces

1. The behaviour of the KafkaProducer client will be updated so that the Callback.onCompletion method will be called by the producer in a manner consistent with the Javadoc. This means that we will no longer call Callback.onCompletion with a null value for metadata, which currently happens in one part of the code base (see here).

Proposed Changes

1. In the one case where KafkaProducer is calling Callback.onCompletion directly, update the call to pass a placeholder value for metadata (in a manner consistent with the Javadoc) instead of a null value. The placeholder value would be as follows:
    a. TopicPartition: Topic would always be correct. Partition is either a valid partition value (if known) or '-1' otherwise.
    b. All other fields (offset, timestamp, serializedKeySize and serializedValueSize) would have a value of -1 explicitly set.

A tested implementation of this change can be found here, which includes some refactoring and changes to unit tests to try to ensure more consistent handling of callbacks.

Compatibility, Deprecation, and Migration Plan

1. For any users who depend on metadata always being null in Callback.onCompletion whenever an exception occurs (which was the documented behaviour until Kafka v.2.2.0), this is a breaking change. Metadata was always null (as per documentation) until Kafka v.1.1.0, when a breaking change was already accidentally introduced in one part of the code as part of PR #4188 (KAFKA-6180) that assigned a placeholder value to metadata instead of propagating the null value. Any users expecting metadata to always be null using a later version than Kafka v1.1.0 will already be experiencing unexpected errors and issues in their usage of the client whenever metadata is not null, although this change would increase the likelihood of those issues occurring.
2. Any users who depend on metadata always being a placeholder value and never null (as per documented behaviour since Kafka v2.2.0), may be experiencing unexpected errors and issues in their usage of the client when metadata is null, and this change will fix that issue for those users.

3. I believe it is best to simply make the breaking change that aligns all calls to Callback.onCompletion with the documented behaviour and to try to communicate this to users. Although it is a breaking change, I believe it is safer than the current situation where there is an inconsistency between documented behaviour and implementation that may be currently causing issues for users.

Rejected Alternatives

1. Updating the Javadoc to document the current behaviour for Callback.onCompletion instead of updating the behaviour to align with the Javadoc was considered. However, the documented behaviour is consistent with the behaviour for Interceptors, so this approach was rejected as it would result in less consistency in the behaviour of the Callback and Interceptor APIs.
2. Continuing to use Callback instead of InterceptorCallback inside clients.producer.internals would result in less overall changes, while still fixing the current issue with the behaviour of Callback.onCompletion. However, this would prevent us from being able to test the expected behaviour in an easy way (this is assuming that 'clients.producer.internal' Interfaces can be updated without considering it a major breaking change).