KIP-467: Augment ProduceResponse error messaging for specific culprit records

- Status
- Motivation
- Public Interfaces
- Proposed Changes
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives

Status

Current state: *Accepted*

Discussion thread: link

JIRA: KAFKA-8729 - Getting issue details... STATUS

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

Motivation

Today when a ProduceRequest containing multiple partition data is received, the following validation logic will be executed before appending the data to corresponding partition logs:

- Messages has valid magic number compared with its outer record batch; if not throw **InvalidRecordException**.
- Messages for compacted topics must have keys; if not throw **InvalidRecordException.**
- When magic value >= 1, messages must have monotonically increasing (relative) offsets starting from 0; if not throw **InvalidRecordException.**
- When magic value >= 1 and validate that timestamp Type is CREATE_TIME; and also message's timestamp is within the range of configured DiffMaxMs. If not throw **InvalidTimestampException.**
- When magic value <= 1, validate message CRC (for magic > 1 there's no record-level CRC); if not validate throw **InvalidRecordException**.
- When magic value >= 2, check that record batch has valid offset range, count, sequence number and are not control records; if failed throw **InvalidRecordException.**
- For transactional / idempotent record batch, also validate the following:
  - Configured magic number should >= 2, if not throw **UnsupportedForMessageFormatException.**
  - Producer epoch should be larger than or equal to book-kept epoch, if not throw **ProducerFencedException**.
  - If producer epoch is larger than book-kept epoch, check sequence is 0; if not throw **OutOfOrderSequenceException** or **UnknownProducerIdException** depending on epoch.
  - If producer epoch is equal to book-kept epoch, check that sequence is continuous; if not throw **OutOfOrderSequenceException** or **UnknownProducerIdException** depending on sequence number.
  - **NOTE** that **OutOfOrderSequenceException** can only be thrown from the callback, while **UnknownProducerIdException** can be thrown directly from the caller of send() / commitTxn() etc as well.

And the above exceptions would cause the whole batch (and therefore the whole partition data) to be rejected with the corresponding error code – note that the only exception is **InvalidRecordException**, which inherits from CorruptRecordException and hence would result in **CORRUPT_MESSAGE (2)** which is a retriable error, but many of those **InvalidRecordException** cases above are actually not-retriable at all. All other error codes correspond to an APIException and hence thrown to user's callbacks / Future object directly.

However, a lot of those errors above are actually triggered by a single record, not at the record-batch level; but nevertheless when the whole batch was rejected, and all record's Future callback will throw the same exception which is very confusing (think: a send() call of record B failed because another record A is corrupted, but the same exception would throw for record B's callback indicating corrupted error). So we'd like to 1) introduce more information in the returned error message of the ProduceResponse to improve such cases; also 2) introduce a separate error code from the retriable **CORRUPT_MESSAGE** which indicate fatal errors from invalid record.

Public Interfaces

We propose to add the following new fields into the produce response:

```
Produce Response (Version: 8) => [responses] throttle_time_ms
  responses => topic [partition_responses]
    topic => STRING
    partition_responses => partition error_code base_offset log_append_time log_start_offset
      partition => INT32
      error_code => INT16
      base_offset => INT64
      log_append_time => INT64
      log_start_offset => INT64
      error_records => [INT32]         // new field, encodes the relative offset of the records that caused
error
      error_message => STRING          // new field, encodes the error message that client can use to log itself
    throttle_time_ms => INT32
```

Also a new error code:

```
INVALID_RECORD(85, "Some record has failed the validation on broker and hence be rejected.",
InvalidRecordException::new);
```

Proposed Changes

1. Let **InvalidRecordException** to not inherit from CorruptedException anymore, instead inherit from ApiException directly (which is non-retriable). And also moved it to "org.apache.kafka.common" to become a public class.
2. On the broker side:
    a. For the above cases which throws **InvalidRecordException** that indicates fatal errors (i.e. except the case of CRC checksum failures which we would change the error code to CORRUPT_MESSAGE), return the new error code INVALID_RECORD.
    b. When setting the error code, if there are multiple types of errors within a single batch, since for most cases we will throw the exception right away, we would only indicate one error code which would be the first error encountered while validating the batch.
        i. For that error code to set, we will try to encode the list of error_records as relative offsets of the records that are causing the whole batch to be rejected (again, in most cases this would be empty since we throw immediately after the first error).
        ii. For that error code to set, optionally try to encode the customized error message (in most cases it would be empty).
3. On the client side, augment the error handling so that:
    a. If the error_code's corresponding exception is re-triable, follow the current behavior to retry the whole batch as-is (so far the only case would be **CorruptedException**);
    b. If the error_code's corresponding exception is not re-triable, check if error_records is empty or not:
        i. If it is empty, reject the whole batch and set the exception for all the records' future (e.g. **UnsupportedForMessageFormatException** or **ProducerFencedException**).
        ii. If it is not empty, only remove those records in the field, and then retry by creating a new batch with those error records removed (for idempotent producers, also reset the sequence number as well as offset). In this way, records in the same batch would not be rejected as a whole, but some records may still succeed while those culprits be rejected (since this KIP cases would include **InvalidRecordException** and **InvalidTimestampException**).
4. Improve the metrics on broker-side for better user visibility for different types of errors that log-validator would expose under "BrokerTopicStats":

```
-- NoKeyCompactedTopicRecordsPerSec: counter of failures by compacted records with no key
-- InvalidMagicNumberRecordsPerSec: counter of failures by records with invalid magic number
-- InvalidMessageCrcRecordsPerSec: counter of failures by records with crc corruption
-- NonIncreasingOffsetRecordsPerSec: counter of failures by records with invalid offset
```

Compatibility, Deprecation, and Migration Plan

- For old client, instead of returning the error code of **CorruptRecordException** which is concrete but incorrect, brokers would return a different error code for **InvalidRequestException**, which is more general than InvalidRecordException but is at least not mis-leading.
- Old versioned broker would not be a problem since client can still handle all the existing error code normally.

Rejected Alternatives

None.