

KIP-724: Drop support for message formats v0 and v1

- [Status](#)
- [Motivation](#)
- [Public Interfaces](#)
 - [Apache Kafka 3.0](#)
 - [Apache Kafka 4.0](#)
- [Proposed Changes](#)
 - [Apache Kafka 3.0](#)
 - [Apache Kafka 4.0](#)
- [Compatibility, Deprecation, and Migration Plan](#)
 - [Apache Kafka 3.0](#)
 - [Apache Kafka 4.0](#)
- [Rejected Alternatives](#)
- [Future Work](#)
 - [Remove v0 and v1 support on the read path as well](#)



Status

Current state: Adopted

Discussion thread: [link](#)

Vote thread: [link](#)

JIRA:

-  Unable to render Jira issues macro, execution error. (3.0)
-  Unable to render Jira issues macro, execution error. (3.1)
-  Unable to render Jira issues macro, execution error. (4.0)

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

Motivation

Message format v2 was introduced in Apache Kafka 0.11.0 (released in June 2017) via [KIP-98](#) and has been the default since. It includes a number of enhancements (partition leader epoch, sequence ids, producer ids, record headers) required for correctness ([KIP-101](#), [KIP-279](#), [KIP-320](#)), stronger semantics (idempotent producers, transactional clients) and other features ([KIP-82 - Add Record Headers](#), [KIP-392: Allow consumers to fetch from closest replica](#)).

Four years later, it's time to sunset message formats v0 and v1 to establish a new baseline in terms of supported client/broker behavior and to improve maintainability & supportability of Kafka. This also aligns with [KIP-679](#), which will enable the idempotent producer by default in Apache Kafka 3.0 (and requires message format v2). We propose the deprecation of message formats v0 and v1 in Apache Kafka 3.0 and to disallow writes with v0 or v1 in Apache Kafka 4.0.

Public Interfaces

Apache Kafka 3.0

1. `log.message.format.version` & `message.format.version` are deprecated, a warning is issued if the value is lower than 0.11.0 and it is always assumed to be 3.0 if the `inter.broker.protocol.version` is 3.0 or higher (see below for the implications).

Apache Kafka 4.0

1. `log.message.format.version` & `message.format.version` are removed. They won't serve any purpose and the fact that the allowable values are Kafka versions instead of message format versions has been a source of confusion. If we introduce new message format versions, they should use actual message format versions (v0, v1, v2, v3, etc.).
2. Produce requests v2 or lower and up-conversion from message formats v0 and v1 won't be supported.
3. Fetch requests v3 or lower and down-conversion to message formats v0 and v1 won't be supported. However, v0 and v1 records may be returned if they are stored on disk in such message format versions. This is necessary to avoid scanning through all record batches before they are returned. See "Future Work" section for possible options for avoiding this in the future.

Proposed Changes

Apache Kafka 3.0

Once a broker is upgraded to 3.0 and the `inter.broker.protocol.version` is updated to 3.0, `message.format.version` is assumed to be 3.0 and we will write records with message format v2 in the following scenarios:

1. Persisting produce records on disk
2. Persisting group data (consumer offsets and group metadata)
3. Writing new segments as part of log compaction

1 and 2 are straightforward since that's the current behavior when the message format is explicitly set to 3.0, but 3 introduces new behavior. The main goals are to use v2 for all client writes and to opportunistically convert to the new format where there is little downside. We will discuss subtle correctness considerations later in the document.

Note that followers will continue to write the records they receive from leaders without conversion. Since leaders will use v2 for new requests, replicating old data is the only case where v0 or v1 records may be written to disk after the upgrade to Apache Kafka 3.0 and the `inter.broker.protocol.version` is 3.0.

Produce and fetch requests with v0 and v1 message formats will be supported via up-conversion and down-conversion. Up-conversion and (especially) down-conversion have measurable performance impact due to increased CPU and memory usage, but the vast majority of Kafka clients have supported v2 for some time (even Spark, a notable late adopter, has supported v2 since Spark 2.4, which was released in October 2018).

The current heuristics to detect whether a down-conversion is required rely on an additional scan to check if there are messages with a newer message format than the max version required by the fetch request. With the changes proposed in this KIP, we expect messages on disk to have format version 2 in the common case. Given that, the heuristics are no longer useful and we will always down-convert if the fetch request is v3 or lower.

There are subtle correctness considerations when up-converting records that had been previously written with an old format. Replication is aligned by batches, but up-conversion will happen independently on each broker for case 3. For compressed records, v0 and v1 message formats support batching, so it's straightforward. Uncompressed records, however, are not batched in v0 and v1. To ensure alignment, we will convert them to single record batches in v2. It's worth noting that message format v2 is slightly less efficient than v0 and v1 when single record batches are used, but it's an acceptable cost for correctness and it only impacts older records. Over time, these records will either be deleted via retention or will be replaced by new versions for compacted topics.

Apache Kafka 4.0

We will remove write support for message formats v0 and v1 in Apache Kafka 4.0. Consumers will continue to support message formats v0 and v1. Similarly, brokers will still return message formats v0 and v1 with fetch v4 or higher.

Compatibility, Deprecation, and Migration Plan

Apache Kafka 3.0

As described, produce requests from producers with no v2 message format support will require up-conversion while fetch requests from consumers with no v2 message support will require down-conversion. To avoid negative performance impact, we recommend upgrading to newer versions (anything released in the last 2 years should be fine, although some clients may require configuration not to use ancient protocol versions).

Apache Kafka 4.0

Clients with no support for message format v2 will not be supported. In the rare cases where such clients are still used, they will have to be upgraded. Fetch request v4 or higher is required for message format v2, so fetch v3 and older would no longer be supported by the broker or Java consumer. Similarly, Produce request v3 or higher is required for message format v2, so produce request v2 and older would no longer be supported by the broker or Java consumer.

Rejected Alternatives

1. Maintain support for message formats 0 and 1. Message format 2 is required for correctness (KIP-101) and key features like idempotence and transactions (KIP-98).

Future Work

Remove v0 and v1 support on the read path as well

There are two aspects to this change: broker and consumer.

On the broker, we can achieve this by providing a mechanism to force up-convert all the record batches on disk. For example, we introduce a new config `log.record.version.force.upgrade` with two possible values: `null` (default) and `2`. If the value is set to `2`, the broker ensures all segments have records with format `v2` after log recovery has completed during start-up. This can be extended to support newer message format versions if and when they are introduced. This config is only allowed to be non-null if the `inter.broker.protocol.version` is 3.0 or higher. In a subsequent major release, we would drop support for reading `v0` and `v1` records from disk.

On the consumer, it's more complicated. Brokers assume that consumers can read all message format versions and brokers never up-convert records when handling fetch responses. Furthermore, we cannot change the behavior of older broker versions. The options here are a bit complex and require more thought.