KIP-597: MirrorMaker2 internal topics Formatters

Status

Current state: Accepted

Discussion thread: *here*

JIRA: *KAFKA-10232*

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

Motivation

MirrorMaker2 uses a few internal topics to store some of its metadata. Records in these topics use binary formats so they can't be directly displayed but first need to be parsed and formatted. A way to easily parse and format records in these topics would be handy when investigating issues.

Kafka already has the concept of MessageFormatters used for other internal topics using binary format. We have OffsetsMessageFormatter, GroupMetadataMessageFormatter, TransactionLogMessageFormatter and the console consumer tool is able to use them to dump the content of __consumer_offsets for example.

The proposal is to provide similar Formatters for the MirrorMaker2 topics: checkpoints, heartbeats and offset-sync. At the same time, I propose publicly exposing a MessageFormatter interface that replaces the internal MessageFormatter trait we currently have. That way the new Formatters can be in the mirror project.

Additionally, having a public interface will also enable users to build their own formatters that can be reused with the console-consumer tool. For example, one could create a formatter that works with a schema registry, or a formatter that hides some fields based on the user identity.

Public Interfaces

1) New MessageFormatter interface

It makes sense to reuse the existing MessageFormatter interface. However at the moment, it's not public and it is in the core project. I propose making a new interface public in the org.apache.kafka.common package in clients:

```
package org.apache.kafka.common;

import java.io.PrintStream;
import java.util.Properties;

import org.apache.kafka.clients.consumer.ConsumerRecord;

/**
 * This interface allows to define Formatters that can be used to parse and format records read by a
 *  Consumer instance for display.
 * The kafka-console-consumer has built-in support for MessageFormatter, via the --formatter flag.
 *
 * Kafka provides a few implementations to display records of internal topics such as __consumer_offsets,
 * __transaction_state and the MirrorMaker2 topics.
 *
 */
public interface MessageFormatter extends Configurable, Closeable {

    /**
     * Initialises the MessageFormatter
     * @param props Properties to configure the formatter
     * @deprecated Use {@link #configure(Map)} instead, this method is for backward compatibility with the
older Formatter interface
     */
    @Deprecated
    default public void init(Properties props) {}

    /**
     * Configures the MessageFormatter
     * @param configs Map to configure the formatter
     */
    default public void configure(Map<String, ?> configs) {
        Properties properties = new Properties();
        properties.putAll(configs);
        init(properties);
    }

    /**
     * Parses and formats a record for display
     * @param consumerRecord the record to format
     * @param output the print stream used to output the record
     */
    public void writeTo(ConsumerRecord<byte[], byte[]> consumerRecord, PrintStream output);

    /**
     * Closes the formatter
     */
    default public void close() {}
}
```

2) MirrorMaker2 formatters

The 3 formatters will be in a new package named org.apache.kafka.connect.mirror.formatters in the mirror project:

- HeartbeatFormatter: `> ./bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic heartbeats --formatter org.apache.kafka.connect.mirror.formatters.HeartbeatFormatter --from-beginning`

  ```
  Heartbeat{sourceClusterAlias=B, targetClusterAlias=A, timestamp=1588502119726}
  ```

- CheckpointFormatter:

  ```
  > ./bin/kafka-console-consumer.sh --bootstrap-server localhost:9093 --topic A.checkpoints.internal --
  formatter org.apache.kafka.connect.mirror.formatters.CheckpointFormatter --from-beginning

  Checkpoint{consumerGroupId=qwert, topicPartition=A.heartbeat, upstreamOffset=631, downstreamOffset=631,
  metatadata=}
  ```

- OffsetSyncFormatter:

```
  > ./bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic mm2-offset-syncs.B.internal --
formatter org.apache.kafka.connect.mirror.formatters.OffsetSyncFormatter --from-beginning

    OffsetSync{topicPartition=heartbeat-0, upstreamOffset=0, downstreamOffset=0}
```

Proposed Changes

1) kafka.common.MessageFormatter

This old trait will be deprecated and changed to extend the new MessageFormatter interface. This will allow users to keep their custom implementation and easily transition to the new interface. Finally, the trait will be deleted in the next major release.

kafka.common.MessageFormatter

```
package kafka.common

/**
  * Typical implementations of this interface convert a `ConsumerRecord` into a type that can then be passed to
  * a `PrintStream`.
  *
  * This is used by the `ConsoleConsumer`.
  */
@deprecated("This class is deprecated and will be replaced by org.apache.kafka.common.MessageFormatter.",
"2.6.0")
trait MessageFormatter extends org.apache.kafka.common.MessageFormatter {
}
```

2) Update existing Formatter to use the new interface

Existing Formatters:

- DefaultMessageFormatter
- LoggingMessageFormatter
- NoOpMessageFormatter
- OffsetsMessageFormatter
- GroupMetadataMessageFormatter
- TransactionLogMessageFormatter

Compatibility, Deprecation, and Migration Plan

Existing MessageFormatters implementations will require no changes beyond recompilation. This allows users to transition to the new interface before the trait is deleted in the next major release.

Rejected Alternatives

- Provide a tool or a new mechanism to format binary topics: While this may not require add a new class in the public API, it would not be consistent with the tools users and administrators are already used to.
- Keep the Scala MessageFormatter interface to retain full compatibility: This seems excessive to keep both interfaces as it was not part of the public API and the interface was not mentioned in the console-consumer output.