KIP-128: Add ByteArrayConverter for Kafka Connect

Status

Current state: Accepted

Discussion thread: *here*

JIRA: *here*

Released: 0.11.0.0

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

Motivation

Kafka Connect is designed to handle copying structured data between Kafka and other systems. Handling **structured** data is important since it is often necessary to perform translations and transformations between systems, and in many cases moving data to/from another system doesn't even make much sense without structure (e.g. copying data from a relational database). This is one of the reasons Kafka Connect is an additional layer on top of the otherwise format-agnostic Apache Kafka (serializers are available in the clients for convenience, but in no way affect the rest of the system, most especially not the brokers).

However, there are cases in Kafka Connect where dealing with the raw data is ideal. A couple of examples are:

- Mirroring data: an exact copy is desired, so deserializing/converting the data to Connect format is wasteful
- Output systems/formats that will just copy the raw data: for example, copying each record as a single line into a JSON file if it is already JSON encoded. Or copying what is truly a binary BLOB from a database into Kafka and from there to HDFS.

To address these use cases where avoiding the cost (CPU, memory, garbage collection overhead), this KIP proposes providing a `ByteArrayConverter` supporting only a single type to be converted to/from the Kafka Connect data API, much like the `ByteArraySerializer`/`Deserializer` already provided for Java clients.

Similar classes are already being implemented by third parties in their own connectors, so providing a standard implementation would avoid unnecessary duplication of effort.

Public Interfaces

The new public interface is a class implementing the `Converter` interface that works only with `byte[]` data. The implementation is trivial enough to be included inline in the KIP:

```
package org.apache.kafka.connect.converters;

import org.apache.kafka.connect.data.Schema;
import org.apache.kafka.connect.data.SchemaAndValue;
import org.apache.kafka.connect.errors.DataException;
import org.apache.kafka.connect.storage.Converter;

import java.util.Map;

/**
 * Pass-through converter for raw byte data.
 */
public class ByteArrayConverter implements Converter {

    @Override
    public void configure(Map<String, ?> configs, boolean isKey) {
    }

    @Override
    public byte[] fromConnectData(String topic, Schema schema, Object value) {
        if (schema != null && schema.type() != Schema.Type.BYTES)
            throw new DataException("Invalid schema type for ByteArrayConverter: " + schema.type().toString());

        if (value != null && !(value instanceof byte[]))
            throw new DataException("ByteArrayConverter is not compatible with objects of type " + value.
getClass());

        return (byte[]) value;
    }

    @Override
    public SchemaAndValue toConnectData(String topic, byte[] value) {
        return new SchemaAndValue(Schema.OPTIONAL_BYTES_SCHEMA, value);
    }

}
```

Proposed Changes

We propose adding the new `ByteArrayConverter` interface. The implementation is trivial, but there are a few key characteristics worth noting:

- There should be no configuration values. None should be required since the data is being passed directly through to the Connector.
- When converting to the Connect data API, the schema is `Schema.OPTIONAL_BYTES_SCHEMA`. This provides support for `null` values so this `Converter` can be used with compacted topics.

Kafka Connect divides its classes into multiple jars; especially for the provided JSON converter this is done to allow removing the converter and all of its dependencies. This Converter will be added directly to the `runtime` jar since it introduces no dependencies, is broadly useful, and doesn't warrant its own jar.

Compatibility, Deprecation, and Migration Plan

Adding `ByteArrayConverter` has no implications for existing classes, compatibility, deprecation, or migration.

Test Plan

The general Converter interface is already well tested by Connect unit and system tests. Given its simplicity, a small number of unit tests should sufficiently cover this new class.

Rejected Alternatives

None