KIP-206: Add support for UUID serialization and deserialization

Status

Current state: *Accepted*

Discussion thread: here

JIRA: *KAFKA-4932*

Released: 2.1.0

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

Motivation

Some users use a UUID (Universally unique identifier) as the key for Kafka messages. However since Kafka has no built-in UUID Serializer / Deserializer, UUIDs cannot be used out of the box and they need to be converted either to a String or to a byte[]. Having UUID Serializer / Deserializer built-in would make this easier as it will allow producers to set UUID objects directly as the key, and have it automatically deserialized in consumers.

Proposed Change

This KIP proposes adding new UUIDSerializer and UUIDDeserializer classes as well as support for the new classes into the Serdes class. This will allow using UUIDs directly from Consumers, Producers and Streams.

UUID serialization and deserialization will be done through String into a 36 bytes array (and the other way around for deserialization). The String representation of UUID is common across platforms and programming languages. (See Rejected Alternatives as to why the String representation is used and not the Binary representation)

Public Interfaces

- New class `org.apache.kafka.common.serialization.UUIDSerializer` which implements the `Serializer<UUID>` interface
- New class `org.apache.kafka.common.serialization.UUIDDeserializer` which implements the `Deserializer<UUID>` interface
- New method `static public Serde<UUID> UUID()` in `org.apache.kafka.common.serialization.Serdes` class
- New subclass `UUIDSerde` in `org.apache.kafka.common.serialization.Serdes` which creates a new serde based on the `UUIDSerializer` and `UUIDDeserializer` classes

Migration Plan and Compatibility

This KIP is a new implementation and doesn't have any backwards compatibility issues or special requirements on migration from older versions.

Rejected Alternatives

UUIDs can be also represented in binary format as 16 bytes array. However, different platforms / programming languages have different interpretations of the binary format (little endian, big endian etc.). Because of this, it seems reasonable to support only the String representation which has a chance to keep a good compatibility between different clients without the need to configure the encoding format.