KIP-713: Validation of Enums in configuration

Status

Current state: Under Discussion

Discussion thread: *here [Change the link from the KIP proposal email archive to your own email thread]*

JIRA: *KAFKA-12301 [Change the link from KAFKA-1 to your own ticket]*

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

Motivation

Enums are used throughout the Kafka codebase and are heavily utilized in the Kafka Connect ecosystem. Supporting parsing and validation of enum values would provide a better user experience by utilizing error messages that have a clear corrective action. For example take a typo while configuring `security.protocol` throws the following error message.

```
Exception in thread "main" org.apache.kafka.common.KafkaException: Failed to create new KafkaAdminClient
        at org.apache.kafka.clients.admin.KafkaAdminClient.createInternal(KafkaAdminClient.java:479)
        at org.apache.kafka.clients.admin.Admin.create(Admin.java:61)
        at org.apache.kafka.clients.admin.AdminClient.create(AdminClient.java:39)
        ...
Caused by: java.lang.IllegalArgumentException: No enum constant org.apache.kafka.common.security.auth.
SecurityProtocol.SASL_PLAINTEXTA
        at java.lang.Enum.valueOf(Enum.java:238)
        at org.apache.kafka.common.security.auth.SecurityProtocol.valueOf(SecurityProtocol.java:26)
        at org.apache.kafka.common.security.auth.SecurityProtocol.forName(SecurityProtocol.java:72)
        at org.apache.kafka.clients.ClientUtils.createChannelBuilder(ClientUtils.java:103)
        at org.apache.kafka.clients.admin.KafkaAdminClient.createInternal(KafkaAdminClient.java:454)
        ... 7 more
```

The first question a user would ask is what is a valid value? Let me search for the docs. A better user experience would be to throw a more descriptive error message. The error message should give the user enough context to attempt to correct the issue without reading the docs.

```
Exception in thread "main" org.apache.kafka.common.KafkaException: Failed to create new KafkaAdminClient
        at org.apache.kafka.clients.admin.KafkaAdminClient.createInternal(KafkaAdminClient.java:479)
        at org.apache.kafka.clients.admin.Admin.create(Admin.java:61)
        at org.apache.kafka.clients.admin.AdminClient.create(AdminClient.java:39)
        ...
Caused by: org.apache.kafka.common.config.ConfigException: Invalid value SASL_PLAINTEXTA for security.protocol.
Enum value not found. Valid values are: PLAINTEXT, SASL_PLAINTEXT, SASL_SSL, SSL
        at java.lang.Enum.valueOf(Enum.java:238)
        at org.apache.kafka.common.security.auth.SecurityProtocol.valueOf(SecurityProtocol.java:26)
        at org.apache.kafka.common.security.auth.SecurityProtocol.forName(SecurityProtocol.java:72)
        at org.apache.kafka.clients.ClientUtils.createChannelBuilder(ClientUtils.java:103)
        at org.apache.kafka.clients.admin.KafkaAdminClient.createInternal(KafkaAdminClient.java:454)
        ... 7 more
```

The second version of this error message gives the user a message that is specific to the version of software they are using. It read the enum that is present in the JVM and outputs the available constants for the enum.

Public Interfaces

The changes to the API consist of two parts. The AbstractConfig class is modified to include methods for working with Enums. The second part are changes to the ConfigDef classes that will provide recommended values and validation.

```
class AbstractConfig {

  private static <T extends Enum> T parseEnum(String key, String value, Class<T> enumType) {
    try {
      return Enum.valueOf(enumType, value);
    } catch (IllegalArgumentException ex) {
      String message = String.format(
          "Enum value not found. Valid values are: %s",
          Stream.of(enumType.getEnumConstants())
              .map(Enum::name)
              .collect(Collectors.joining(", "))
      );
      throw new ConfigException(name, value, message);
    }
  }

  public <T extends Enum> List<T> getEnums(String key, Class<T> enumType) {
    List value = getList(key);
    List<T> result = new ArrayList<>();
    for (Object entry : value) {
      result.add(parseEnum(key, value, enumType));
    }
    return result;
  }

  public <T extends Enum> T getEnum(String key, Class<T> enumType) {
    String value = getString(key);
    return parseEnum(key, value, enumType);
  }
}

class ConfigDef {

  public static class ValidEnum<T extends Enum> implements ConfigDef.Validator {
    final Set<String> validEnums;
    final String message;

    public static <T> ValidEnum<T> of(Class<T> enumClass, T... excludes) {
      return new ValidEnum(enumClass, excludes);
    }

    private ValidEnum(Class<T> enumClass, T... excludes) {
      Set<T> exclude = new HashSet<>(excludes);
      this.validEnums = Stream.of(enumClass.getEnumConstants())
          .filter(e -> !exclude.contains(e))
          .map(Enum::name)
          .collect(Collectors.toSet());
      this.message = this.validEnums.stream()
          .sort()
          .collect(Collectors.joining(", "));
    }

    @Override
    public void ensureValid(String s, Object o) {
      if (o instanceof String) {
        if (!validEnums.contains(o)) {
          throw new ConfigException(
              s,
              String.format(
                  "'%s' not found. Valid values are %s.",
                  o,
                  enumClass.getSimpleName(),
                  message
              )
          );
```

```
        }
      } else if (o instanceof List) {
        List list = (List) o;
        for (Object i : list) {
          ensureValid(s, i);
        }
      } else {
        throw new ConfigException(
            s,
            o,
            "Must be a String or List"
        );
      }
    }

    @Override
    public String toString() {
      return this.message;
    }
  }

  public static class EnumRecommender<T extends Enum> implements ConfigDef.Recommender {
    final Set<String> validEnums;

    public static <T> EnumRecommender<T> of(Class<T> enumClass, T... excludes) {
      return new EnumRecommender(enumClass, excludes);
    }

    private EnumRecommender(Class<T> enumClass, T... excludes) {
      Set<T> exclude = new HashSet<>(excludes);
      this.validEnums = Stream.of(enumClass.getEnumConstants())
          .filter(e -> !exclude.contains(e))
          .map(Enum::name)
          .collect(Collectors.toSet());
    }

    @Override
    public List<Object> validValues(String s, Map<String, Object> map) {
      return Collections.unmodifiableList(validEnums);
    }

    @Override
    public boolean visible(String s, Map<String, Object> map) {
      return true;
    }
  }
}
```

Compatibility, Deprecation, and Migration Plan

Additional methods and classes will be added to ConfigDef and AbstractConfig. Kafka Connect plugin developers utilizing this functionality could receive a MissingMethodException if their plugin is used with an older Connect Worker.

Rejected Alternatives

One possibility was to extend ConfigDef.Type to include ENUM as a type similar to CLASS. This would require extensive modification to the ConfigDef classes because it doesn't currently support a way to specify the type of CLASS or ENUM. This is validated later. Providing a Validator and method to read from a string requires less modification while achieving acceptable results.